Title       An Online Sharable Diary System using Sun Technologies
Name        Roger Farrow

This is a digitised version of a dissertation submitted to the University of Bedfordshire.

It is available to view only.

This item is subject to copyright.
Roger Farrow
ID number 0708955

Thesis Title: An Online Sharable Diary System using Sun Technologies
MSc Thesis

Course: MSc Computer and Internet Applications
Department of Computer Science & Technology

Supervisor: Dr. Paul Sant

2009/10
Abstract

Title: An Online Sharable Diary System using Sun Technologies

The need for an online sharable diary came from a company who were having trouble keeping track of their employees during the day using a traditional paperbased diary solution, they had tested a variety of software solutions (Outlook and iCal) but these solutions didn't gel well across the variety of operating systems employed by the company. So a solution was sought that was free of operating system dependency yet robust and secure.

The technologies chosen for this implementation were those owned by Sun and include OpenSolaris, Netbeans, MySQL and GlassFish. With Servlet and JSP being the intended programming and scripting approaches. This was for a variety of reasons but predominantly because they are available free, are enterprise level software, and because Sun offers comprehensive support options. Initial research was collected from the company in the form of questionnaires and diary samples and these contributed heavily to the development.

The methodology chosen for this implementation was prototyping which worked well and enabled communication between the client and developer on a variety of issues, it helped the client feel more involved and that they would receive a useful product at the end.

The implementation undertaken suffered a wide range issues both software and hardware that resulted in the program not being completed successfully. This caused changes to be made in both the development process and for the developer to have to favour JSP over Servlet for programming approach. However the HTML, CSS, database, logo and branding components were completed. An unexpected but positive outcome was also the development of a five bit binary style access control system, that reduced the number of database records and interactions required.

Whilst it is impossible to state that this project was a success for either the client or the developer. The client although they are still looking for an adequate solution were interested in the development process undertaken as a means to producing criteria that would inform a decision on a solution they were also interested in the combination of use of Google calender and Doodle as means to share diary events, and to smooth the organisation of events. The developer found the taking of client requirements and using them to shape the development process was very informative, and the use of Prototyping was successful and contributed to good communication between the client and developer, there were also significant learning points relating to testing and establishing a development environment. The unexpected benefit of the creation of a binary style access control was also interesting although more development is required.
Acknowledgements

I would like to thank Paul Sant for his advice, Des Stephens for his encouragement, Caroline Farrow for her unwavering support and the client for providing this opportunity. Thank you all.
THESIS AUTHOR CONSENT FORM

AUTHOR’S NAME: ROGER FARROW

TITLE OF THESIS: AN ONLINE SHARABLE DIARY SYSTEM USING SUN TECHNOLOGIES

DEGREE: MSC COMPUTER AND INTERNET APPLICATIONS

Please read carefully and sign the following as appropriate.

I have read and understood the University’s regulations and procedures concerning the submission of my thesis.

I understand that I have already signed a declaration agreeing to my dissertations being kept in the Learning Resources Centre (LRC) when I enrolled.

We would like now, to extend this agreement by making the thesis available online. Further to this,

I AGREE AS FOLLOWS:
- That I am the author of the work.
- That I have exercised reasonable care to ensure that the Work is original, and does not to the best of my knowledge break any UK law or infringe any third party’s copyright or other Intellectual Property Right.
- The LRC and BREO administrators do not hold any obligation to take legal action on behalf of the Depositor (you), or other rights holders, in the event of breach of intellectual property rights, or any other right, in the material deposited.

DELETE ONE OF THE FOLLOWING OPTIONS AS APPROPRIATE:

1. I hereby extend my consent to this thesis being included in the LRC as well as on BREO via online access.

2. I hereby DO NOT extend my consent to this thesis being included in the LRC as well as on BREO via online access.

AUTHOR’S PERSONAL SIGNATURE:

AUTHOR’S STUDENT NUMBER: 0708955

DATE: 26. 02. 2010
<table>
<thead>
<tr>
<th><strong>Name</strong></th>
<th>Roger Farrow</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ID</strong></td>
<td>708955</td>
</tr>
<tr>
<td><strong>Programme</strong></td>
<td>Computer and Internet Applications</td>
</tr>
<tr>
<td><strong>Mode of study</strong></td>
<td>F/T</td>
</tr>
<tr>
<td><strong>Project title</strong></td>
<td>An Online Sharable Diary System using Sun Technologies</td>
</tr>
<tr>
<td><strong>Supervisor</strong></td>
<td>Dr. Paul Sant</td>
</tr>
<tr>
<td><strong>Date:</strong></td>
<td>24/02/10</td>
</tr>
</tbody>
</table>
Contents

1) Introduction: 7
   1.1) Problem 7
   1.2) Main Aim 7
   1.3) Methodologies and Outcomes 7
   1.4) Thesis Structure 8

2) Research: 11
   2.1) Background 11
   2.2) Specific Studies 11
   2.3) Technologies 11
   2.4) Case Studies 13

3) Questionnaire Results 16

4) Design, Development, and Implementation 19
   4.1) Requirements 19
   4.2) Design and structural information 21
   4.3) Data Acquisition 34
   4.4) System Implementation 36
   4.5) Testing and Analysis 45
   4.6) Improvements 50

5) Overall Evaluation 51

6) Conclusions and Recommendations 54

7) References 56

8) Appendices 60
   Appendix I: The Project Questionnaire 61
   Appendix 2: The structural diagrams in sequential order 62
   Appendix 3: The Project Poster 83
1) Introduction

Within this section we will consider the 'problem', which is the issue that motivated the need for this project to be undertaken by the developer for the client. We will also consider the main aims, these are the key goals that the client has intimated to the designer that should be reached. The methodology is the testing and communicating ideology, that will be used during the project the pros and cons of which are discussed. Finally the thesis structure lays out the order of the report and sparsely explains each section.

1.1) Problem:

The problem extends from a company who have found difficulties in finding a diary solution that assists in forward planning and allows them to view business engagements of other employees so that they can accurately determine where they are. The company uses a variety of operating systems (MacOS and Windows) so many traditional software options have failed to be fully compatible and stable.

To this effect they are interested in a web based solution that will allow the account owner to add or delete appointments, invite colleagues to events and view colleagues appointments.

The solution must also incorporate access control, because of the sensitive nature of the store data, that shares information between colleagues and chosen recipients, but also allows them to black out private appointments.

1.2) Main Aim

The main aim of this project is to produce a web based diary solution which is sharable with colleagues that uses Sun technologies (for example OpenSolaris OS, GlassFish Webstack, and JavaEE, EJB, JSP and servlets).

Objectives:

1) Develop an online Java based diary management system that allows users to add, view, and edit diary entries.

2) Develop a platform independent solution that is compatible with a wide range of operating systems and web browsers.

3) Produce an ergonomic intuitive GUI with some customisation options.

1.3) Methodology

For this project the chosen approach was Prototyping. This methodology was chosen for several reasons, firstly the client, although heavily involved in design and project management are not familiar with software development, and have a level of uncertainty about their needs. In this situation Prototyping approaches can help to clarify and
determine the requirements. Similarly where a graphics interface is required, a version of it can be produced and the interaction and level of appeal can be assessed this can be an advantage with forming a positive impression of the product, which in this situation will be necessary because the client whilst technologically adept is conservative about the adoption of new systems. There are also concerns with the company about uptake and usage, Prototyping methodology helps the client and the designer, to work more closely together, which should hopefully improve their enthusiasm for the product because they feel genuinely involved in the engineering process (van Vliet, ch 3.2.1, 2008).

However the issues of this approach that must be avoided are unnecessary features, which will lead to scope creep and make the product unsuccessful and unwieldy. There may also be issues relating to the level of experience with the developer both technically and with familiarity with this methodology. Both of these issues are highlighted by van Vliet, as areas that can be potentially disadvantageous but as the author advises an awareness of these issues enables them to be guarded against (ch 3.2.1, 2008).

1.4) Thesis Structure

1) Introduction:
   1.1) Problem:
   Looks at the background of the need by the company to develop a shareable online diary

   1.2) Main Aim
   This section looks at the key goals and objectives that need to be achieved to call the project a success.

   1.3) Methodologies and Outcomes
   Prototyping is the selected methodology for this project and in this part the advantages and disadvantages of this methodology are considered.

   1.4) Thesis Structure
   Gives a brief overview of the structure and short description of the content in each of the thesis sections

2) Research:
   2.1) Background
   Defines the use of the term diary in the context of this project, and how it will shape the product.

   2.2) Specific Studies
   Considers the clients issues, and the necessity of security when handling data that is as private as that intended to be handled by the product.

   2.3) Technologies
   Critiques the software intended to be used for development environment by comparing it to similar offerings, before finally considering the strengths of using Sun's products as a stack because of Sun's support options.
2.4) Case Studies
Similar products to the project's product are critiqued to see if there is a case to continue development of the product.

3) Questionnaire Results
In this section the questionnaire and diary sample results are analysed and discussed, and prepared so that they can contribute to the rest of the project.

4) Thesis:
4.1) Requirements
This sets out the components that are to be considered essential features against which a the project can be deemed to have succeed or not.

4.2) Design and structural information
This is where the different steps of the design stage were considered including the UML and other diagrams, and the design of the access control.

4.2) Data Acquisition
The client's contribution to the project, in this case questionnaires, product name, navigation, and iteration plan are discussed here.

4.3) System Implementation
This part is broken into several sections that reflect the different parts of the development process. These include: Layout and graphics user interface, form construction, database construction and program implementation

4.4) Testing and Analysis
A testing plan is described here, before being used to analyse the products.

4.5) Improvements
Here the project is analysed to see what improvements could be made both to this project and how this project's issues can be used to inform future projects.

5) Overall Evaluation
In this section the project as a whole is evaluated by looking at the different sections that make up the project and then finally assessing it against the requirements list.

6) Conclusions and Recommendations
In this section we consider, what both the client and developer have received from the project, and look at learning points, and potential areas for future development.

7) References
Here is a comprehensive list of all the resources used throughout this project.

8) Appendices
   Appendix I: The Project Questionnaire
   Appendix 2: The structural diagrams in sequential order
   Appendix 3: The Project Poster
Within the Introduction chapter we have consider firstly the 'problem' that this project aims to address, we have collected the project's main aim and objectives, as these will be key in the design process. The methodology is the next section and obviously Prototyping was selection it was also stated how this methodology should assist the development process. Before finally describing how this thesis will be laid out.
2) Research:
The research was conducted as a literary review, which begins with the 'background', this section aims to clarify what is meant by the term 'diary' as it used within the context of this report. The 'specific studies' section focuses on the issues the client has faced whilst trying to find suitable software and the need for security in a site that handles this kind of sensitive data. 'Technologies' forms the next part within this section the different components of the development environment are considered and discussed and critiqued against rivals, to show why they were selected for this project. The final section for this chapter is 'case studies' which considers the intended project product against similar competitors, and software and critiques them to establish that there is a business case for the product.

2.1) Background

A key area to stress is the difference between a diary or day book, which is for recording key dates and future events and as such is for forward planning, and a journal or blog which is predominantly aimed at recording and reflecting upon past events, rather than planning future ones. Internet blogging or journal tools are very common and widely used such as http://www.xanga.com/ however the focus of this project is on producing a product that can be used to assist forward planning whilst also enabling colleagues to know the whereabouts of each other.

2.2) Specific Studies

Through discussions with the company the biggest issue that has been encountered by the company in their search for a suitable application has been finding software that works and integrates across the whole network. The easiest way to avoid these issues is to use an online application that is accessible through a web browser as this deals with the issues of cross operating system issues.

Because of the nature of the data being stored within and used by the application there will have to be appropriate security measures implemented to protect the privacy of the users of the application and ensure the safety of their data. Also the data will have to be shared carefully to that effect a system of user groups or access control lists will have to be implemented.

2.3) Technologies

OpenSolaris 2009.6

An Enterprise grade Unix based operating system intended for a wide range of users and purposes, despite being a free open source operating system has support options (Crimson Consulting Group, p3, 2009). This operating system is based on Sun's Solaris operating system and is geared at giving users access to features before they are incorporated into Solaris (Subramanyam, Smith, and Zhang, p5, 2009). An OpenSolaris
development environment also enables a smooth transition to an enterprise Solaris serving environment at a future point. As a long term solution OpenSolaris has many features that recommend it, including its wide support for modern and legacy hardware, 32bit and 64bit, single and multi-core chips (Solter, Jelinek, and Miner, p5, 2009), it also has its own file system called ZFS which scales to cope with large data volumes particularly well, (Solter, Jelinek, and Miner, p223, 2009) and it has integrated web services. (Solter, Jelinek, and Miner, p5, 2009).

MySQL 5.1

Database software, is a open source solution which is owned by Sun, its nearest competitors are Oracle and Microsoft solutions both of which are proprietary. Another strength is that it is included in the bundled download produced by Sun (NetBeans, 2010), and is free and can be supported by Sun as part of their support services, reducing the complexity for a company of having multiple support numbers. As a long term solution MySQL should be a cheaper option as it boasts better reliability, performance and scaling and that it will be at least 90% less expensive to run and license than its proprietary competitors (Sun Microsystems Inc., p9, 2009). MySQL can also reduce long term hardware costs by not requiring proprietary hardware, and by having a "superior performance on lower cost hardware" (Sun Microsystems Inc., p10, 2009). MySQL is also well known and well tested, and is a component of the ubiquitous LAMP web stack which is widely used for web development (Sun Microsystems Inc., p3, 2009). These features make MySQL an ideal database solution for a new project that has the potential to become a long term investment where cost will be an initial concern and scalability and reliability will become increasingly important.

GlassFish v2

A collection of web containers produced and distributed by Sun it supports Java Enterprise Edition (J2EE) and a wide range of other features as well as servlets JSP (Java Server Pages) (Sun Microsystems Inc., p7, 2008) and it is intended to be an industrial grade product for providing dynamic web applications. The closest well known product to GlassFish is Tomcat, which is a Java container intended for providing support to servlets and JSP obviously GlassFish because it supports Java enterprise edition which allows for greater flexibility in programming approach. GlassFish also demonstrates superior performance (Sun Microsystems Inc., 2010) and a much wider range of features to Tomcat (Sun Microsystems Inc., p7, 2008) as well as a superior administration and management tool (Sun Microsystems Inc., p4, 2008) amongst other features all of which make GlassFish a superior application environment to Tomcat.

GlassFish V2 has better support for Netbeans than V3 currently, and is also the most tested and reliable of the three versions (Java.net, 2008) it is also JavaEE certified which V3 is not yet (Sun Microsystems Inc., p7, 2008).

NetBeans IDE 6.7.1

NetBeans is Sun's IDE, its competitor is Eclipse, both have excellent integration with GlassFish but the integration between GlassFish v2 and NetBeans (Java.net, 2008) is very
strong and well documented compared with Eclipse. However the really strong feature of NetBeans is the combined downloads that Sun offers to make installing and integrating a more straightforward process. These combined downloads include GlassFish, NetBeans and MySQL (Netbeans, 2010) and are significantly easier to use than repository downloads or worse mixing repository and internet downloads (Leonard, 2009)

**PHPMyAdmin**

PHPMyAdmin is an administration tool for maintaining MySQL databases, enabling a suitably permissioned person to interact with and modify databases set up new users and execute statements all through a very convenient interface (PhpMyAdmin, 2010, (A)). PHPMyAdmin has been consistently winning awards since 2003, including Sourceforge's best tool for SysAdmins 2009 (PhpMyAdmin, 2010, (B)). As such its usefulness as a MySQL tool is hard to dispute. PHPMyAdmin is run through Apache 2.2 this is the only reason for having Apache in this stack.

**Why Sun products compared to any others?**

The Sun server stack and OpenSolaris/Solaris is an attractive enterprise grade offering of products supported comprehensively by a single organisation for a subscription. Compared to traditional LAMP and Tomcat solutions which are inconsistently supported if at all. A Crimson Consulting Group white paper (2009) compares support offerings from Sun with Red Hat, because of the two companies similar product lines and support options. The paper checks a wide range of details, and find that Sun has an edge in their offerings compared to those of Red Hat, because of level, competency, response time and cost of support service, and because they have options that support OpenSolaris. These factors make Sun products a very attractive option to any enterprise wanting to start a highly reliable web service.

2.4) **Case Studies**

**Traditional Diary:**

A book, for writing in appointments and logging key dates. Adair states that a diary is an 'essential tool for time planner' (p57, 1987) and are available in a wide variety of forms and durations, and recommends the format of keeping a large page per day diary in the office, as these are cumbersome to move about, and a pocket diary on your person then synchronising these upon return to the office (p57-59, 1987).

However Adair has already given us aspects of diary use that can be interpreted as negatives, firstly the need to maintain multiple diaries to keep adequate track and details about daily engagements. This introduces the potential for errors if the synchronising isn't accurate. Also there is the need to replace the diary when its duration has expired and of course to synchronise the previous diary with the current one and if you are running multiple diaries to synchronise both of them to avoid double booking scenarios (Adair, p58, 1987).

This project aims to avoid these aspects because the solution could reduce the need to use multiple diaries and synchronise them across years, and daily. Adair's (p57-59, 1987)
diary description also details the functions of a diary and as such contributes to features that need to be considered during the development.

**Outlook:**

Microsoft's email and diary program forms part of office and is commonly used across a homogeneous business computing environment for various diary functions, including logging important dates and appointments, checking other peoples diaries and inviting people to meetings (Microsoft, 2010 (A)). This solution however is unavailable for the Mac portion of their network until Office 2010 is released (Microsoft, 2010 (B)) . Prior to this Microsoft called Outlook for macs Entourage however the company stated they had issues getting this software to run across the Macs and issues again integrating it with Outlook on their Windows PCs.

**Doodle:** [http://www.doodle.com/](http://www.doodle.com/)

Currently a popular way to create meetings by creating a poll for a meeting it enables invited people to vote for preferred dates and times (Doodle AG, 2010). As such it enables an individual to organise an event without having to call every person they want to invite it also enables people to select a time that suits them without being put on the spot.

Doodle is good for organising individual meetings but doesn't enable you to plan more than a single event per poll or to maintain a running list of appointments such as is the goal of this project, it does have ever provide an excellent solution to the issue of coordinating meetings with people external to your organisation.

**Google Calendar:**


Google's online application that enables people to create and share calendars. Google's approach is the most similar to the goals of this project, Google's application rests on the idea that you can produce and manage multiple calendars and import them from associates and organisations either on a by individual basis, or by importing a public calendar (Google, p1-5 2010, (A)). Likewise you can create and publish calendars and events on the same basis either to specific people or as a public file(Google, 2010, (B)). This application enables a variety of views including day, four days, month and agenda and enables you to toggle the various calendars you have overlaid on and off(Google, p4 2010, (A)). Adding events is very intuitive and easy (Google, p1 2010, (A))and that is something to aspire to in this project.

However there is a lack of pre-arranged access control which could make sharing a little more tricky, and the privacy options are also different to those planned for this project and the privacy options planned for the project fit a closer with the companies concerns about security and simplicity of use.
iCal:

iCal came to our attention after the questionnaire responses were collated, it is software for Mac OS X and is intended as a piece of calendar/diary management software that enables you to manage events and categorise them into separate calendars so you can split up or view the total of your commitments. iCal also gels with Mac OS X's Mail and Address Book software, so you can take event invites directly from Mail, and put them into iCal, and let iCal use your Address Book, to dispatch invites and log replies if both users have iCal and Mail (Apple, 2010). However for the company in question iCal, is not supported by Windows Operating Systems, and has also failed to gel across their Mac systems, meaning that it is not a consistent solution.

This closes the Research chapter, over the course of this chapter we have narrowed and clarified the meaning of 'diary' so that it fits our project, and consider the key issues of compatibility, and the product needing to be operating system independent. We have critiqued the software and the support package provided by Sun, and found their offerings to be of high quality and very appropriate for an enterprise setting. We have also assessed whether there is a development case for the product by evaluating similar competitors, and it has been determined that this project would offer something more in each case.
3) Questionnaire Results

For the questionnaire please see Appendix I.

The questionnaire covered four key areas that would feed directly into the project, these areas are, diary use, events, access control, and online application use, also collected were samples of the potential user's paper diaries so that the usage could be directly analysed. These samples were collected by taking two random weeks from the responder's 2009 diary.

These samples and the questionnaires themselves could only be collected if confidentiality was insured, and as long as the diary samples were taken from previous months, as a result of this names and specific details will not be addressed in this report, nor will the actual questionnaires that were received be included. However the specific details are irrelevant in this context because the study was to examine the general usage of diaries rather than the individuals specific usage.

From the general diary use section we found that all the the responders used diaries, and that the majority of them used them for work events, while only half used them for home family or personal appointments. Additional categories that were suggested were birthdays and holidays, both of these were of significant interest because birthdays can be broadened to annual events, and holidays, are of a period of time that people may wish to include events within e.g. Flight times and holiday excursions. The frequency that the diary was used to record events was considered to be daily for business uses, while only weekly for personal, friends, and home events and the sorts of events that were recorded under each category are recorded in the table (table 1) below:

<table>
<thead>
<tr>
<th>Event type</th>
<th>Business</th>
<th>Home</th>
<th>Personal</th>
<th>Friends</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Meetings</td>
<td>Vets</td>
<td>Holiday</td>
<td>Dinners</td>
</tr>
<tr>
<td></td>
<td>Visits</td>
<td>Significant others shifts</td>
<td>Dinners</td>
<td>Parties</td>
</tr>
<tr>
<td></td>
<td>Holiday</td>
<td></td>
<td>Drinks</td>
<td>Cinema</td>
</tr>
<tr>
<td></td>
<td>Events</td>
<td></td>
<td>Birthdays</td>
<td>Going out</td>
</tr>
</tbody>
</table>

The events component of the questionnaire focused on information recorded and the setting up of an event. In terms of the information stored about an event, all the responders recorded the appropriate times and names of attendees, with other pieces of information that maybe stored including short descriptions, clients company, travel time, and meeting headline, eg specific product. The responders were split evenly over whether they would record different information for different events. The results for the previous two questions were complicated by the responses to 'How would what you record differ if the appointment was in your office compared to visiting their premises?' with the majority recording location and primary client contact details, while others recorded travel time and
'here' or 'at' to make location clearer. When specifically asked the majority of responders did record travel time in their record.

We also queried how an event was arranged, the majority of people used email, but phone, 'in person' and by being 'invited' where other responses. When asked what would make organising events easier the responders mentioned iCal (Apple, 2010) but requested a version of it that 'worked' and was not operating system dependant, they also mentioned a calendar that would allow the sending of invites and attachments, and anything that would reduce the number of emails going 'back and forth'.

Next we addressed the issue of access control, events of a personal nature were just to just the user and specific individuals, clients should be treated as individuals, while work engagements should be shared to all colleagues, so that all members of the team are aware of that persons commitments. Amongst examples of inappropriate sharing of events/information were sharing internal business information to external people/clients, and personal information to colleagues and clients, with the necessity to separate home and private life from work and clients.

The final section of the Questionnaire dealt with online application use, the most commonly used sites by the responders are, Facebook (www.facebook.com) Google (www.google.com), hotmail (www.hotmail.com), Amazon (www.amazon.co.uk) and the BBC news (www.bbc.co.uk). The reasons for these sites being so popular were cited as 'easy to use' and social networking capabilities. However other reasons included 'simple to log out', fast, 'good clarity' and 'clear images'. The factors that encouraged repeated use were the popularity, and common use of the site, and that it is an easy way to contact people, and that its 'fun' these are clearly key factors in relation to social networking sites. In relation generically to sites, there were 'helpful', 'up to date', 'easy links' and 'customisable'.

The final comments section of the questionnaire drew these responses, that an online service should be clear, simple, functional, without user issues, and up to date, and that it should not be operating system dependant.

The analysis of the diary samples involved listing all the events recorded eg. deadline for X company, meeting with X clients, X person's birthday. Then simplifying the events in that list so that the events were expressed in generic terms, eg. deadlines, meetings, and birthdays, and then categorising them under the classifications of work, friends, family and personal. Please see table 2.
Table 2: The classification of events found in the diary samples.

<table>
<thead>
<tr>
<th>Type of Event:</th>
<th>Personal</th>
<th>Family</th>
<th>Friends</th>
<th>Business</th>
</tr>
</thead>
<tbody>
<tr>
<td>Events:</td>
<td>Trips</td>
<td>Special occasions</td>
<td>Trips</td>
<td>Staff Reviews</td>
</tr>
<tr>
<td></td>
<td>Concerts</td>
<td>Holidays</td>
<td>Concerts</td>
<td>Holidays (others and own)</td>
</tr>
<tr>
<td></td>
<td>Doctors</td>
<td>Dog care</td>
<td>Birthdays</td>
<td>Appointments</td>
</tr>
<tr>
<td></td>
<td>appointments</td>
<td>Trips</td>
<td>Anniversaries</td>
<td>Visits by clients</td>
</tr>
<tr>
<td></td>
<td>Beauty</td>
<td>Dog care</td>
<td>Anniversaries</td>
<td>Visits to clients</td>
</tr>
<tr>
<td></td>
<td>Treatments</td>
<td>Trips</td>
<td>Special occasions</td>
<td>Extended visits</td>
</tr>
<tr>
<td></td>
<td>Holidays</td>
<td>Birthdays</td>
<td></td>
<td>(eg work experience)</td>
</tr>
<tr>
<td></td>
<td>dog care</td>
<td>Anniversaries</td>
<td></td>
<td>Shows</td>
</tr>
<tr>
<td></td>
<td>Birthdays</td>
<td>Special occasions</td>
<td></td>
<td>Lunch</td>
</tr>
<tr>
<td></td>
<td>Anniversaries</td>
<td></td>
<td></td>
<td>Deadlines</td>
</tr>
<tr>
<td></td>
<td>Special occasions</td>
<td></td>
<td></td>
<td>Occasions</td>
</tr>
</tbody>
</table>

This was predominantly to test that every generic event in the diary samples would fit under one at least one classification.

The final step of the analysis was too look at the information that was recorded about each event, and to further generalise the event types. The key information that was recorded for events was time, location, duration, type, and some individual information. While the most generalised event types were, annual occasions, holidays, deadlines, and appointments.

To conclude this chapter the value of this data and how it has informed each area of development is invaluable. By trying to find the commonalities and the differences, in the way that people use their diaries will shape the site because it has increased our understanding of what the client wants from the product and also the data that the site and database need to be able to handle.
4) Design, Development, and Implementation:

Within this chapter, we consider the design, development and implementation of the project, starting this the 'requirements', where we consider the basic needs of a site and combine those with the needs of the client to provide a list that can be used to guide design, and check the level of success at the end of the project. We look at the 'design and structural information' where the information that has been collected and requirements are used to produce a range of UML diagrams and other diagrams, to map the functions and layout of the site. The 'data acquisition' part of this chapter looks at the information that was contributed to by the client and their involvement in varying aspects of the site including, questionnaires, name, navigation and methodology. The system Implementation section looks at the development of the product in a step by step fashion it also looks at the issues that were experienced during the process and significantly impaired the completion of the product. This then lead to the test and analysis stage of the process for which a testing list was established to check the components and functions against the testing could only be partly completely due to the program being incomplete. This then leads into the improvements section where areas that could have been improved upon. As such attention is paid to Design Issues, Development Environment Issues, Development, and the Access Control system.

4.1) Requirements

The requirements for this project are primarily based from the aim and the objectives which were: to make a web based diary solution using Sun technologies, that incorporates an easy to use graphic user interface (GUI) with customisation options, is operating system and browser independent and allows users to add view and modify diary entries.

However the results from the questionnaire highlight other features that should be included, including the need to control access to the user's data, make the site as simple to use as possible, easy to read and easy to understand, the questionnaire also provided a list of possible categories, event types, and exemplary websites to look at and incorporate into the design process. The questionnaire also proved the necessity of having access control for user data, but also that it would be necessary to split the section of business into colleagues and clients and that because clients potentially all work for different organisations it is necessary to treat them as individuals.

There is also the requirement that there is security in place to safe guard the personal nature of the information being manipulated by the application.

Once the clients requirements had been collected and the questionnaires analysed, it was necessary to compose a list that lays out their requirements with the standard requirements, which can be used through out the design and implementation process and finally to assess the level of success against.

Requirements list:

1) Functions:
Registering
Recovering password
Logging in
Adding an Event (including ACL Levels)
Deleting an event
Modifying an event
Finding other users
Adding people and assigning ACL levels
linking a person to an event
Deleting a user account
using PhPMvMyAdmin for Databases and user accounts
Log out

2) Appearance
   Graphics user Interface
      intuitive and straightforward navigation.
      a customisable feature
      clear and easy to read

3) Security
   Access control system
      Suitable data and account protection methods

4) Other
   Appropriate Database
   CSS controlled HTML
   Information and Help pages
   Appropriate Testing
4.2) Design and Structural Information

The design of this project required many different components that would all support each other. The database, needs to be formed to hold the requirements of the site and to actively link all the components together. There needs to be forms that will collect and manage the appropriate information. Access control system needs to be developed that is complex enough to protect the information, whilst being simple enough for the user to take fullest advantage of. The graphics user interface, needs to be simplistic yet enough to cover the spectrum of functionality, and also needs to incorporate an element customisation. Whilst the programs and their functions need to be organised, and modelled.

This process started by analysing the functionality that would be involved, this was done using a ‘use case diagram’ please see fig 1 (for a sequential view of all the structural diagrams please view appendix 2, for fuller versions of all UML diagrams please see the attached disk (MscProject09-10 > diagrams ) for both image and Visual Paradigm files or MscProject09-10 > thesis > ThesisFigs to have them in numbered order. ). The Use Case diagram was based on the functions from the requirements list created in the previous chapter. This Use Case Diagram has formed part of the back bone of the project, and has been used to establish the Activity, Sequence, and Class diagrams, its also fed into developing the website Page and Function Summary, and the Graphic User Interface once the different use cases and actors had been established and grouped in terms of function e.g. all the ‘event’ components together.

The uses cases are described within the diagram in the sparsest possible terms to keep it as simple as possible, there are 2 types of customer an unregistered user and register user, the unregistered user only needs access to register and to information about the product, when the user becomes registered they then inherit the access to these use cases and also to the Log in use case. To demonstrate that there is a necessity to log in all the other application use cases extend from Log in. On the employee side there would be community managers, who try and assist people in times of need as such they have all access to the use cases as the registered customers, but they also have limited database access. While the super user would be key members of the development team and heads of the community team who could be trusted and may need even less restricted access to the database, for setting up other database users or making changes to the database its self.

Within the Use Case Diagram, there are two use cases that were not developed these were the 'information' pages, and the staff database access ('log in', 'Access database' and the 'database access secure features'), the information pages were left aside as no programming was intended for them as they are intended to be a collection of html pages intended to provide end user level explanations of usage. Whilst the staff database access was due to be handled by PHPMyAdmin as such there is no need to create diagrams to create that aspect of the product.
Figure 1: Use case diagram for the Diaryshare Project
Fig: 2 Activity diagram of Registering

- open form
- complete form
- submit form
- correctly filled
- incorrectly filled
- produce error
- generate error message
- contact database
- database related error
- query unsuccessful
- complete query
- return message
- query successful
However once the use case diagram was generated it was necessary to create the activity diagrams, to chart the logical steps in each use case, please see Figs 2-12 in appendix 2 and Figs 2, 5, and 12 have been incorporated into the text for easy illustration but with their original numbers so their context can be placed in the sequential diagram order of appendix 2.

The activity diagrams follow a common thread that they start at the top and flow downwards, errors and failures either return to the user to an earlier stage in the diagram or terminate the process prematurely. They illustrate the breaking down of a process into logical steps and decisions that are then used to inform the sequence diagram, or can be used to program that particular activity.
Within the activity diagrams we tried to illustrate security aspects that could be incorporated, such as JavaScript form checking (fig 5) to reduce the possibility of incorrect data being entered, or possibly SQL injection techniques being employed. While Figs 5 and 12 incorporate the need to resubmit the user's security details before access to modifying the user's account is permitted. This to prevent intentional or accidental damage to the user's access when the user has failed to log out. However to have these levels of security on every transaction would make the application too time consuming and cumbersome to be usable. So the all the other transactions, with the exception of registering and logging in, require an active and valid session. These Activity Diagrams were used to assess that all the logical steps were present in the functions.
The next stage was the development of the sequence diagrams (Figs 13-18 in appendix 2, diagrams 16 and 18 are included above), these are intended to map the interactions between the different layers of the program, in this case they focus on the user, the application and the database tiers. Breaking the activities into the different tiers also forces the developer to consider at what level they want which parts of activities to take place.

The Sequence Diagrams also grouped the functions that were dealt with individually in the Activity Diagrams for example Fig: 18 combines adding an event, assigning access control and linking people to that event, modifying, and deleting an event, because these functions all act on the same type of data.

Both sequence diagrams (Fig: 16 and 18) incorporate 'if' options these demonstrate when the program has to make a decision, for example if the form has been correctly filled or not, if it hasn't been correctly filled the program forwards a request to the user for more information, it is then up to the user if they decide to correct the information and resubmit or leave the process at that point.

The next development stage was the Class Diagram (Fig: 19) this was to organise the different objects and find the variables and the different processes that each object would perform, this also helped to identify the relationships between those objects. The processes in each object came from the original Use Case Diagram, and the objects were suggested by the Sequence Diagrams. While the variables came from a combination of the questionnaire results, and necessity imposed by the normalisation of the data base which then fed into the Database Entity Relationship.
Fig: 19 Class Diagram

Event
- UserID (owner)
- eventID
- time
- date
- location
- travel time
- additional information
- recipients user ids [ ]
- event acl details [ ]

Add event
Modify event
Delete event
get user id
check session
Set Event ACL

These 3 arrays need to be linked together and incorporated into a link list rather than an array

Messages
- message ID
- user ID (owner)
- user ID (receiver)
- date
- time
- title
- content

inbox
sentbox
get user ID
get recipient ID
compose
read message

Registration
- user ID
- first name
- second name
- password
- personal email
- business email
- bg colour

Registration form
Create registration
Delete registration
modify registration

Log
- email (main)
- password

create session
check login details
end session

<<bean>>

Session
- session duration
- user ID
- session validity
- bg colour

return login details
return user id
return session details

Home
- user ID
- page information
- session response
- bg colour

Members
- user IDs and acls [ ]
- user ID (associate)
- user ID (owner)

find person
associate person
disassociate person
set acl for person
get user ID
send message
get acl for person

create session
check login details
end session

Members
- user IDs and acls [ ]
- user ID (associate)
- user ID (owner)

find person
associate person
disassociate person
set acl for person
get user ID
send message
get acl for person

create session
check login details
end session

Registration
- user ID
- first name
- second name
- password
- personal email
- business email
- bg colour

Registration form
Create registration
Delete registration
modify registration

Log
- email (main)
- password

create session
check login details
end session

Roger Farrow 0708955
Database Entity Relationship Diagram (Fig: 20) was the next step after the normalisation of the questionnaire results and the addition of other essential features, such as personal details for creating an account from, and automatically incremented fields for use as primary keys. In this diagram each box represents a table, the text above the box is the table name, and the text within the box is the fields, the primary key is in bold. The relationship between the tables is also shown.

The final diagram that was created is the Site Page and Function Summary (Fig: 21) this diagram is to illustrate the flow of the site, and to analyse the different components of the site so that the navigation can be designed. It also serves to show the navigation that is necessary in each section, and it does it in a very concise and visual form. The diagram forms a way to understand how the processes described in the class diagram are
distributed in relation to the pages in the site.

The next area of planning was to plan the access control, as this is a key aspect of the site because it will be the basis of the sharing of the events that individuals have input. The initial idea was to ascribe each event, and each person to person association with an individual access record, however there were concerns about the number of records this would generate. So a way to simplify and reduce the number of records was sought. The solution was that since there five parts of the access control system these parts could be treated as a five bit binary calculation, which would provide a finite number of access levels, in fact thirty one of them, and the primary key would be the value of the calculation, for an example of this please see Table: 3.

Table: 3 An Example of the Access Control Levels Calculations.

<table>
<thead>
<tr>
<th>Primary Key</th>
<th>Clients</th>
<th>Colleagues</th>
<th>Friends</th>
<th>Family</th>
<th>Personal</th>
</tr>
</thead>
<tbody>
<tr>
<td>Binary Value</td>
<td>16</td>
<td>8</td>
<td>4</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>30</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>31</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

The final section of planning was to consider the graphics user interface, the links aspect of this were planned in Fig: 21, but other design aspects had to be considered such as the width of the site. In the past the usual width for a site was around 750 pixels, this was because 800x600 pixels was considered to be the lowest common denominator of screen resolutions, so a fixed site width could be assured to fit well on the widest number of possible resolutions. However in recent years the quality and availability of higher resolution, cheaper and wide screen monitors has significantly increased meaning that the number of 800x600 resolution machines was 'steadily shrinking' (Niederst Robbins, p36, 2006). However with the recent interest in 'netbook' style laptops and smart phones, there is a need to support smaller resolutions again, with netbooks such as the Asus eee701 having a screen resolution of 800x480 pixels (Ried, 2007) however more recent versions have incorporated larger resolutions. Except that an increasing number of smart phones are also equipped with 800 pixel resolution screens as well as browsers that are capable of handling normal HTML/XHTML rather than reduced HTML or WML and JavaScript this makes it necessary to start considering smart phone screen resolutions (Anon, 2010 and Beavis, 2010). Consequently because of the necessity to make this project as flexible as possible in terms of dependencies it was decided that a traditional width measurement would be the best, so 750 pixels was opted for as this would fit comfortably on the screen of larger smartphones, smaller netbooks, and the majority of desktops and still allow for
the browser's interface which can vary considerably (Niederst Robbins, p28-36, 2006).

Next was the issue of the locations of essential site features, such as branding, navigation and information areas, for this the sites that were mentioned in the questionnaire results were viewed to see what general trends were (Facebook, (www.facebook.com) Google, (www.google.com) hotmail, (www.hotmail.com) Amazon, (www.amazon.co.uk) and BBC news (www.bbc.co.uk)). On these sites branding was uniformly applied in the top left corner sometimes extending across the top of the page. Navigation was also applied on the left side and across the top under the branding and sometimes a little higher on the left side, with informational areas centrally located below the branding. These aspects would be used to shape the layout, so that it would follow a layout that is familiar to the end users. It was also noticed that all these sites used a white background for informational areas, and the majority used black text for majority of paragraph text.

The next consideration was how to incorporate a customisable feature into the user interface, as was noticed from the above research the informational areas should be should be white with predominantly black text, and the main areas were located in familiar areas, which left the background colour, which could be selected during registration from a limited number of colours, but integrating this into the rest of the design was tricky. White was the colour that was going to be predominantly used on the site, so if branding was also white but used an alpha channel such as that provided by Gif files the background colour could be incorporated into the design which would enable the user to customise a key feature of the site to a more personally aesthetically pleasing colour.
Data Acquisition:

The company were very happy to help throughout the project with information and offers of technical and design assistance, however because of the academic nature of this project the offers of technical and design assistance were politely turned down. As a compromise it was offered that, once completed, if the project was successful and application desirable that collaboratively the application could be re-skinned.

Questionnaires

The company allowing me to use questionnaires on their staff, and to include aspects of the results in the design and the report was extremely helpful and played a significant role in shaping and designing the application. The details of this research have been heavily discussed previously.

Name

The company also played a heavy role in selecting a name for the application, with management staff engaging in an over email collaborative effort to compile ideas, this was started by collecting possible terms eg. Diary, Time, Management, before starting to try combinations and ideas, some of the ideas produced were 'Manage Me', 'Tempus Fugit', 'Most Timely' and 'Diary Share' This was eventually narrowed down to just 'Most Timely' and 'Diary Share', before 'Diary Share' was finally selected for its being very descriptive yet short. 'Diary Share' was felt to sum up the project/application perfectly. The client commented that they had enjoyed being involved in this part of the process.

Navigation

The initial navigation and page layout was sent to the company (Fig: 22) and the feedback was that having both an 'in box' and an 'out box' was wasteful and over complicated the navigation and could be replaced with a 'mailbox' option instead (Fig: 23), but that aside from that it was easy to understand and self explanatory as to what each link would do at this stage. It was already clear that Prototyping as a methodology would be extremely beneficial as the client had already made changes at this stage, and it would give them the opportunity to trial the content associated with the links with their perception of what the content should be, so if changes to the content or its organisation were necessary they could be made before the client was presented with a final product.
Establish First Version and Iterative Prototyping:

As the methodology employed for this application development cycle was Prototyping, the intention was to establish a working first version which could be presented to the client to test so that it could be evaluated and feedback could be received and then acted on. Which would hopefully enable a second iteration to be done during the development period of this project. This second iteration would be very time pressured and would depend heavily on the speed that the clients respond with to the first iteration, it would also rely heavily on the first iteration being a largely problem free development.

System Implementation

Layout and Graphics User Interface

The layout and graphics where created in Fireworks 8 by Adobe, (the logo files and the components used to construct it are in the MscProject09-10 > logo folder,) once a logo was produced I started to constructed the layout and site art (the site art and templates are in mscProject09-10 > siteArt folder) so that the the different back ground colours could be tested for suitability (Fig: 24) This also allowed the first iteration of the site appearance to be reviewed by the client (Fig: 22) and using the feedback from them to create the second version of the appearance (Fig: 23). However while the client was reviewing, the art work for the site was sliced into parts ready to be inserted into a CSS controlled HTML layout (for these images please see MscProject09-10 >diary share project html> images> headerparts folder ). The other aspects of the site art and design were created using CSS and HTML, so the change requested by the client to the menu system was made in the HTML documents. Once the site was created in HTML and CSS this would produce an easy template that could be brought into the program and modified according to the pages requirements, for example the index page doesn't require the menu system, while the user home page does.

Form Construction

Forms are a key feature of most web services as they are how the site finds data about the users so that it can be manipulated by the program. However the forms must also relate to the database and be adequate to request sufficient information to complete the database records. The forms components were produced separately from the site templates, so that they could be easily checked and added to the program. The values, form component names, maximum lengths must match the database tables and fields. The form components must also be of the right variety as a tick box form will not be appropriate for collecting a persons email address for example. To avoid these issues the forms were carefully constructed using the Database Entity Relationship Model (Fig: 20) so that the form names would match the database fields they were collecting data for. It was also necessary in the case of the event form to collect data about the access control level, which involved replacing the values with corresponding five bit binary values which would be 1, 2, 4, 8 and 16. However in this case as every event must have personal as its default access control so that the user can view their own events, consequently it wasn't necessary to include the personal access option, so the program would have to anticipate that and add an additional 1 to the form sections value. The form sections that were
created to make generating other parts of the program easier are available from MscProject09-10 >diary share project html folder. An additional consideration when creating these form components was to ensure that they used HTML tags and IDs that had been assigned CSS formatting where appropriate.
Fig: 24 the Logo and Branding with Two Different Background Colours
Database Construction

The database construction and management was predominantly handled by the PHPMyAdmin software which provides a useful interface for MySQL. The tables and fields match the Database Entity Relationship Model (Fig: 20) and incorporate the relevant notes that are in brackets beside the field or table names (Fig: 25, 26). The table called ACLCodes was also filled with the appropriate pre-generated information (Fig: 26).

Program Implementation

The program implementation was undertaken in NetBeans 6.7.1 and was originally begun and intended to be a Java servlet development however soon after the programming began it became obvious that there was a problem as the pre-generated NetBeans Servlet code would not compile due to a 'lost symbol' errors relating to servlet components. This suggested that it was a API issue, the commonest solution for this is to set the 'classpath' environmental variable within the operating system (Sun Microsystems Inc., 2002) once this was set and incorporated the standard java APIs, the Tomcat APIs and the GlassFish APIs, the majority of the issues were resolved. However the same issue persisted with the servlet request API, despite this API being listed within the project resources. This error resisted numerous attempts to resolve it during the time pressured development period it soon became obvious that resolving this issue would consume too much time, so an alternative method was sought. The alternative approach that was selected was to use JSP. Since this point the response API issue has been resolved, this was done by adding the servlet response API to the classpath variable individually rather than the containing folder however this solution was not found until after the development period.

The development commenced using JSP and JSTL which were unfamiliar methods and several resources were used both as reference and informed the development (Brown et al, ch 4 and p149-155, 2001, Brown et al, ch 4, 2003, Giunipero, year unknown, Sun Microsystems Inc., 1999) pieces of script that came from specific sources are commented within the files and full references are included within the references section of this report. Implementation Files are available within the MscProject09-10 > NetbeansProjects > DiaryShare1 folder.

The obvious place to start was with the index file (Fig: 27) and registration file, once these were ready, they were deployed to the GlassFish server and it was obvious there were issues. Firstly it was found that GlassFish doesn't link to pages with a capital letter as the first letter of the file name and also that the CSS wasn't being linked to, this was initially thought to be because the CSS file also started with a capital letter. Replacing capital letters with lowercase resolved the issue quickly, however this didn't resolve the CSS issue, various approaches were tried to resolve this issue but again none seem to yield an appropriate result. These attempts can be seen in the head section of the index.jsp (MscProject09-10 > NetbeansProjects > DiaryShare1> web> index.jsp).

The next issue was the register.jsp (MscProject09-10 > NetbeansProjects > DiaryShare1> web> register.jsp) (Fig: 28) which initially was only showing the background colour and a
The Create Tables within the DiaryShare Database.

Fig: 25 The Created Tables within the DiaryShare Database

- localhost / localhost / DiaryShare1 | phpMyAdmin 2.11.7 - Mozilla Firefox 3.1 Beta 3

- File Edit View History Bookmarks Tools Help


- Page Load... Page Load... Page Load... Setting Up... Java EE &... Creating a... Entities - T... Introduction... Welcome I...

- Most Visited OpenSolaris Sun Microsystems Other Communities

- Page Load... localhost...

- Server: localhost | Database: DiaryShare1

- Structure  SQL  Search  Query  Export  Import  Operations  Privileges  Drop

- Table Action Records Type Collation Size Overhead
- ACLCodes - - x MyISAM latin1_swedish_ci 2.1 KiB -
- Event - - x MyISAM latin1_swedish_ci 1.0 KiB -
- EventPersonLink - - x MyISAM latin1_swedish_ci 1.0 KiB -
- Message - - x MyISAM latin1_swedish_ci 1.0 KiB -
- PersonACL - - x MyISAM latin1_swedish_ci 1.0 KiB -
- User - - x MyISAM latin1_swedish_ci 7.3 KiB 0.8 KiB

- Print view  Data Dictionary

- Create new table on database DiaryShare1

- Name: Number of fields: 

- Open new phpMyAdmin window
### ACLCodes Table Complete with Pre-generated Information

<table>
<thead>
<tr>
<th>ACLCode</th>
<th>personal</th>
<th>family</th>
<th>friend</th>
<th>colleague</th>
<th>client</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>2</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>3</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>4</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>5</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>6</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>7</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>8</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>9</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>10</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>11</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>12</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>13</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>14</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>15</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>16</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>17</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>18</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>19</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>20</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>21</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>22</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>23</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>24</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>25</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>26</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>27</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>28</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>29</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>30</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>31</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
</tbody>
</table>
Fig: 27 Diary Share Index Page.

Welcome to DiaryShare - Mozilla Firefox 3.1 Beta 3

File Edit View History Bookmarks Tools Help

http://localhost:8080/DiaryShare1/

Most Visited
- OpenSolaris
- Sun Microsystems
- Other Communities

Welcome
- phpMyAdmin
- Sun Glass
- Page Load
- Setting Up
- Java EE &
- Creating a...

Diary Share

Please log in:

Please enter your main email address

If you have not registered please click here to do so

Register

For more information about DiaryShare please click here

This site was constructed by WorldWideWhispers 2010
HTTP Status 500 -

Type: Exception report

Message:

Description: The server encountered an internal error () that prevented it from fulfilling this request.

Exception:

org.apache.jasper.JasperException: PW6033: Unable to compile class for JSP

PW6199: Generated servlet error:
string://register.jsp.java:206: 'catch' without 'try'

PW6199: Generated servlet error:
string://register.jsp.java:206: ')' expected

PW6199: Generated servlet error:
string://register.jsp.java:206: not a statement

PW6199: Generated servlet error:
string://register.jsp.java:206: '; ' expected

PW6199: Generated servlet error:
string://register.jsp.java:213: 'finally' without 'try'

PW6199: Generated servlet error:
string://register.jsp.java:217: reached end of file while parsing

PW6199: Generated servlet error:
string://register.jsp.java:218: 'try' without 'catch' or 'finally'

PW6199: Generated servlet error:
string://register.jsp.java:218: illegal character: \26

Note: The full stack traces of the exception and its root causes are available in the Sun GlassFish Enterprise Server v2.1.1 logs.

Sun GlassFish Enterprise Server v2.1.1
As development was progressing it became increasingly obvious that there was a serious issue with the development hardware. Despite the hardware being above the minimum specification for all of the software installed including the operating system, the system was slowing to a crawl and would run out of system resources and require a restart quickly when several pieces of development software were running at the same time. This seriously hampered development and testing. Especially as it became hard to tell if GlassFish was unable to serve a page for technical reasons or because it had run out of resources.

As time was pressing and financial resources were scarce for the project it was impossible to replace the hardware or organise a dual boot environment on another machine. Development had to be separated into longer periods of development and testing. This issues combined with the lack of familiarity JSP, made development a lot more time consuming and unsuccessful. Attempts were made upon the index page, the mailbox functions, registering process, and user’s home (all these and their associated inculdes are in MscProject09-10 > NetbeansProjects > DiaryShare1 > web) before the allocation of time for development expired.

It was intended to get the site running and all the pages present and then begin to implement appropriate security and session information.

**Testing and Analysis**

In organising a testing regime the 'Requirements List' and the 'Use Case' diagram (Fig: 1) was used to identify each function that must be tested, it was also necessary to organise the testing of all other components such as the user interface, and the interaction between the HTML tags and Ids and the CSS. This lead to the establishment of a testing plan.

**Testing Plan:**

1. Site art against different backgrounds in Fireworks
2. HTML and CSS
3. PHPMyAdmin
4. GlassFish
5. testing each page deployed to check it is present
6. each page shows some function

**Use Case based tests:**

7. using PHPMyAdmin for Databases and user accounts
8. Registering
9. Recovering password
10. Logging in

11. Adding an Event (including ACL Levels)

12. Deleting an event

13. Modifying an event

14. Finding other users

15. Adding people and assigning ACL levels

16. Linking a person to an event

17. Deleting a user account

Additional:

18. Log out

19. Testing the event and people ACL levels.

20. Checking the site security

Although the development was severely hampered by the issues explored earlier some aspects of this testing plan could be employed. In fact sections one through seven were used to some degree.

Site art against different backgrounds in Fireworks:
This was tested with all the colours that were selected for use as background colours some of the results for this can be seen in Fig: 24. the tests were successful and the colours that worked best were used.

HTML and CSS:
Both of these features had to be tested to see if they were working adequately and that the CSS would control the layout and formatting of the HTML the results of this can be seen in Fig:23 and 24. This test was also intended to ensure that the HTML link to the CSS page was working, although this failed to ensure it actually worked when the html was made into JSP.

PHPMyAdmin:
Just checking it was present, connecting it to the MySQL program, this was done by looking for the log in page at its default address.

GlassFish:
This was tested starting the domain and by checking the default page displayed on localhost port 8080, port 4848 working was reaffirmed each time a page was tested such
Testing Each Page deployed to Check it is Present
This was done with each page when the project was deployed periodically and was the primary goal behind deploying the project periodically. This was done by checking the page was either producing a result or an HTTP 500 error, rather than the HTTP 404 error which would mean the page didn't exist. The results for this can be seen in Fig: 27, 28, 29 and 30, these results show that the pages that worked only had a limited functionality with some pages showing at least HTML function such as Fig: 27 and 29, while Fig: 28 and 30, show that the page is present but not functioning correctly.

Each page shows some function:
at the stage when development had to be stopped the only page that was showing some function was the index page (Fig: 27), which was processing correctly, and linking to associated pages.

Use case based:

Using PHPMyAdmin for Databases and User Accounts:
This worked well, two accounts were established to remove the need to use the Root account for all access and setting alternative permission levels for the access was a very straight forward procedure. PHPMyAdmin was also used to generate the database and the database tables and fields, these results can be seen in Fig: 25, and 26.

However because the development of the pages never got to the stage where the functions were adequately developed to show some degree of functionality, it was impossible to test whether the use cases were actually addressed, aside from the PHPMyAdmin tests.

To conclude the Design, Development and Implementation chapter, it has to be said that the preparation in terms of the research, and the questionnaire provided an excellent basis for the design phase, which was extremely fruitful. However the Implementation phase started well but quickly ran into a wide range of issues including hardware, and software, that made the development tougher than it should have been. However from this lessons should be taken that it is necessary to test everything even if there complete confidence in it. The result of this oversight is that the product did not proceed well and did not reach its first iteration. However from this chapter the importance of requirement lists, and a thorough testing list for the implementation should be taken forward.
**Fig. 29 The Mailbox Page**

<table>
<thead>
<tr>
<th>Mail Centre - Mozilla Firefox 3.1 Beta 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>File</td>
</tr>
<tr>
<td>---------------</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td><a href="http://localhost:8080/DiaryShare1/mailbox.jsp">http://localhost:8080/DiaryShare1/mailbox.jsp</a></td>
</tr>
</tbody>
</table>

- Most Visited: OpenSolaris, Sun Microsystems, Other Communities

**Enter Email**

- **Home**
- **Add Event**
- **Mailbox**
- **Friends**
- **Help**
- **Edit Account**
- **Log Out**

**Submit**

**Submit**

**Submit**
HTTP Status 500 -

<table>
<thead>
<tr>
<th>Type</th>
<th>Exception report</th>
</tr>
</thead>
<tbody>
<tr>
<td>Message</td>
<td>The server encountered an internal error () that prevented it from fulfilling this request.</td>
</tr>
<tr>
<td>Description</td>
<td>The full stack traces of the exception and its root causes are available in the Sun GlassFish Enterprise Server v2.1.1 logs.</td>
</tr>
</tbody>
</table>

Sun GlassFish Enterprise Server v2.1.1
Improvements

Design Issues:
Upon reflection it is possible to see that two key use cases were left out of the UML stage of the design these were the 'Log out' and 'View Other Users Page' processes, these two stages whilst they didn't impact the level of development, they would eventually have had to be addressed, and without the design phase guidance their development would have been much trickier. However after this experience an improvement would be that the use cases have to be checked more thoroughly against the project goals, and be check for aspects that could be taken for granted.

Development Environment Issues:
The first issue was the classpath variable this could have been improved by testing the environment prior to use. Likewise with the hardware, although it was within the tolerances for the programs should have been tested to see if it would run them satisfactorily as a development environment. There are three clear improvements that could have been
made to this, one would have to been to test the environment more thoroughly, which would then have led to possibly 2 other improvements, which could have been either better hardware or using more efficient programs.

Development:
The development process at the start proceeded well, with the ideas shaping into images CSS and HTML, and making the database, however the actual application development was very troubled, partly because of the hardware constraints that were discussed in the previously, but also by lack of familiarity with JSP and JSTL, this caused significant issues with the project because it slowed development progress considerably. This issues may have been avoided had it not been necessary to change approach to avoid the 'classpath' issues. However this again highlights the improvement of rigorously testing and being familiar with your environment and its limitations before undertaking any project.

Access Control System
The Access Control System was an unexpected discovery that because it wasn't the sole focus of the project didn't get as much development time as it needed to fully develop it. It was very easy to see how to use it to reduce database transactions when the level was created, however it was much harder to know how to reduce the number of transactions when it was in use. This is one area that would benefit from additional development, as well as better use of the persistence tier.

Overall Evaluation
Within this chapter we consider the overall evaluation of the project to this effect we will critique the development on a number of levels, starting with the impact of the methodology on the relationship between the client and the developer and the client and the project.

Prototyping:
Although the project did not come to fullest fruition it did provide an opportunity to use the Prototyping Methodology and explore how it effected the relationship between the developer and the client and the client and the project. As was discussed in the Data Acquisition section, the client enjoyed the level of communication that was encouraged by the methodology, and also the feeling of involvement in the project and the response nature to making changes. Consequently it can be said that this was a successful use of Prototyping, even if the iterations of the final product were not successfully produced.

Software:
OpenSolaris:
OpenSolaris was stable and reliable for the development, there were and the documentation was largely easy to find and understand, so its easy to say that it performed well.

PHPMyAdmin and MySQL:
As a pair these performed extremely well, it was simple to connect the two together and
PHPMyAdmin makes working on and understanding the database much simpler. So these can be considered reliable and valuable pieces of software.

GlassFish:
Was also an excellent piece of software, once it was successfully installed, its performance was good, if hampered by the hardware, the sophisticated and easy to use Admin User Interface (Fig: 31) was a pleasant change from Tomcat or Apache. Another useful and well produced piece of software.

NetBeans:
does well at simplifying the development process once you understand what its doing and how to use that, its very helpful in that it generates a lot of the standard components, file systems and can collate and upload the APIs automatically. It also explains the compiler errors a lot more helpfully than other programming editor software. It combined well with GlassFish, there was however the classpath issue but there is no clear way to know if the fault lay with the classpath or NetBeans.

Hardware:
The hardware was inadequate for the use intended for it, which caused significant issues, so it must be evaluated as being unsuitable for the development and for the software use.

Design:
It was a good experience to take a concept and find sufficient research and information to break the problem down sufficiently to produce the UML and other documentation to be able to see how the product's components and objects would combine to make a full product. However two key components were missed. The design process was also very productive in enabling analysis of the project and did produce a clear map of the way to produce the application. Also aspects of this analysis were of use to the company despite the product being unusable, in working out the key features and functionalities that would allow them to find a suitable solution. On the whole the experience and results have been extremely useful and have genuinely guided this product, to have neglected this phase would have been disastrous.

Implementation Process:

The implementation process started smoothly with site art, HTML, CSS, and the database these components again were essential and it made development much easier to be able to reuse them and know that these components were working as they should be. However the program development itself was very unsuccessful, as not even a first iteration was produced, this is probably the largest weakness with the project.

Requirements list:
The Requirements List, is a key tool for assessing how successful the final product was, started, as can be seen below, with the functions however because of the lack of success with program itself none of the functions worked aside from the PHPMyAdmin function.

1) Functions:
Registering
Recovering password
Logging in
Adding an Event (including ACL Levels)
Deleting an event
Modifying an event
Finding other users
Adding people and assigning ACL levels
linking a person to an event
Deleting a user account
Using PHPMyAdmin for Databases and user accounts
This was tested and used to create the database and additional user accounts

Log out

2) Appearance

Graphics User Interface
This was produced and tested in HTML and CSS and worked, however because the programming didn't work, it wasn't tested properly in context.

Intuitive and Straight Forward Navigation.

The client was satisfied with the HTML and CSS appearance of the second iteration of the navigation. The client also decided that the as far as they could tell the navigation was self explanatory, and therefore straight forward and intuitive.

A Customisable Feature

This feature was to be a customisable background colour, which could be selected from the registration form and modified in the personal details section, this feature was designed and only tested in Fireworks and html, not in the final JSP implementation.

Clear and Easy to Read

To assist with that the site text was black and white and of an approximately 12pt size and the client raised no issue with it.
3) Security
   Access control system

This section was partially designed, but not adequately implemented, but had a lot of potential but it needed more development.

   Suitable data and account protection methods

This section was not implemented aside from the use of password fields, and measures in the UML to safeguard accounts and the necessity of sessions to be implemented.

4) Other

   Appropriate Database

These were produced after normalisation and entity relationship diagram had been completed, and match that diagram. However the databases were never tested to see if they were adequate with the program.

   CSS controlled HTML

This was tested and produced and worked satisfactorily on a browser, however the CSS would not combine with and style the JSP produced html.

   Information and Help pages
These couldn't be formed due to there being no application to document.

In terms of the evaluation of this project, the total was not a success because the implementation of the application was unsuccessful. However there are a considerable number of parts that were successful such as the Research, and the Questionnaires and the way the results from them contributed to the requirements, the design and structure. Also the first steps of the implementation process, the design, the navigation, the HTML, CSS were also successfully completed. However from the use cases and requirements function list only one part was successfully implemented which was the use of PHPMyAdmin for administering the database. Also the components of the implementation that did go well couldn't be tested fully in context, because the later stages of the implementation hadn't been successful. All in all this project cannot be said to have been successful.

Conclusions and Recommendations

In conclusion as had been said previously this project did not successfully produce an adequate product for the client. However the client is making use of the research, and has shown some interest in using Google calendars as a means to keep track of its employees, and Doodle as a way to simplify the organising of meetings, these services where selected as a result of reviewing the criteria and looking at the questionnaire results
consequently for them the project has not been totally unsuccessful.

The developer has taken from this project, more experience of running projects, and collecting data and analysing it so that it can shape design and development. The project has also increased the developer's knowledge of testing and establishing testing and requirement guidelines, as well as encouraging the developer to test the development environment and its hardware far more extensively before using it. There is a need to work on and maintain the familiarity with your chosen development tools and languages. It is also possible that the project was too ambitious for the time scale, and this too is an area that the developer will have to take forward. The methodology and the way it encouraged communication and a good relationship between the client and developer was also an area worthy of note for the developer for future projects.

In terms of future development, the learning points raised above will definitely shape future involvement in projects, there were some especially important lessons relating to testing. As an idea the binary style access control was one that would bear further development, this could be a way to manage various levels of access control with little database interaction. It would also be worth increasing familiarity with JSP scripting. In terms of software the Sun stack and especially GlassFish was very impressive and an opportunity to work with them again would be most welcome.

The strongest recommendation from this project is definitely that testing everything is invaluable.
References


Google (2010) (B) “Share your events with an individual, a group or the whole world” from http://www.google.co.uk/googlecalendar/event_publisher_guide.html viewed on 6.01.10.


Java.net (2008) “GlassFish Project – Netbeans IDE integration” from https://GlassFish.dev.java.net/public/netbeans/ viewed on 06.01.10.


Appendices

Appendix I

the Project Questionnaire as distributed to appropriate company staff, a copy is also within the project files on the submitted CD.

Diary and Online Services Initial Questionnaire V1

Name:
Company:
Contributed diary sheets: Yes/No No of sheets: Contributor number:

Thank you for contributing a sample of your diary, the pages that you have contributed will be analysed to produce statistics relating to the types of data and the client, company, and personal information contained within your diary will remain confidential.

I would like to ask you some questions about your use of diaries and how and what you tend to write in them.

Diary section

Do you use a diary? Yes/No

Do you use it for Work/Home/Family/Personal appointments? (ring as appropriate)

Any additional or irrelevant categories? eg (Work/Home/Family/Personal)

When you write a typical event/appointment into your diary what information do you record about it?

Would you record different information for the different categories of events?

How would what you record differ if the appointment was in your office compared to visiting their premises?

Do you include travel time? Yes/No
How do you arrange an event?

What would make arranging an event easier?

What sort of events do you organise for each of these categories?  
Business, Home, Personal, Friends

How often do you organise events for each of these categories?  
Business, Home, Personal, Friends

If you could share the information in diary how would you want to share the information,

**Online Applications Section:**

Which websites do you use often?

What are the most important aspects of these sites for you?

What are key features that encourage you to use these sites?

Many thanks for your assistance.
Appendix 2
The structural diagrams in sequential order:

Figure 1: Use case diagram for the Diaryshare Project
Fig: 2 Activity diagram of Registering

1. Open form
2. Complete form
3. Submit form
4. Correctly filled
   - Correctly filled
   - Incorrectly filled
5. Generate error message
6. Contact database
   - Database related error
7. Complete query
   - Query unsuccessful
   - Query successful
8. Return message
   - Query successful
Fig: 3 User Log in Activity Diagram

- error message
- present html login form
- complete form
- submit form
- form is checked against database
- Incorrect details
- database error
- success message
- enter user page
Fig: 4 Lost Password Activity Diagram

1. HTML Form for email address
2. Completion of form
3. Contact database
4. Address found
5. Replace password with random one
6. Email password to address
7. Report success
Fig:5 Modify Personal Details Activity Diagram

1. select 'account details' link
2. produce error message
3. request password form
4. password is checked with Database
   - password is incorrect
     - return current details
     - amendments are input
       - deemed incorrect
       - deemed correct
6. confirm amendments
   - deemed incorrect
   - deemed correct
7. input corrections
   - deemed incorrect
   - deemed correct
8. details passed to Database
9. error message
10. details are inadequate
    - details are correct
11. update database
12. success message
13. query failure message
Fig: 7 Find Person Activity Diagram

1. Check session
2. Session is invalid
3. Session is valid
4. Use search
5. Check database
6. Return results
Fig 8 Modify Event Activity Diagram

check session

session invalid

session valid

select existing event

error message

modify details

submit form

generate error

check amendment is valid

amendment is invalid

amendment is valid

contact database

query error

return success message
Fig: 9 Link Person Activity Diagram

- Check session
  - Session invalid
    - Session valid
      - Found person
        - Select individual
          - Select individual's ACL associations
          - Confirm association
          - Email to individual to ask them to confirm association
          - Confirm association
          - Confirm with database
          - Query status
          - Success message
          - Decline association
        - Confirm association
          - Select individual's ACL associations
          - Confirm association
Fig: 11 Deleting an Event Activity Diagram

1. Check session
2. Session invalid
3. Session valid
4. Confirm deletion
5. Deletion denied
6. Deletion confirmed
7. Query goes to database
8. Record deleted
9. Returns message
10. Unsuccessful delete
11. Successful delete

Roger Farrow 0708955
Fig: 12 Deleting a User Registration Activity Diagram

1. Confirm Login
2. Login Invaid
3. Login valid
4. Explain effect
5. Click to remove account
6. Confirm account removal
7. Error message
8. Account removal denied
9. Account removal confirmed
10. Contact database and remove details
11. Query files
12. Return success message
Fig: 13 Registering a New User Sequence Diagram

1. Unregistered User
   1. Complete form
   2. Check form is correct
   3. Form is correct
   4. Return success
   5. Generate success message
   6. Return query success or fail
   7. Fail message
   8. Try again
   9. Return error
   10. Form corrections
Fig: 14 Password Recovery Sequence Diagram

<table>
<thead>
<tr>
<th>Registered User</th>
<th>Site</th>
<th>Database</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input email address</td>
<td>1: check form is correct</td>
<td>2: return result</td>
</tr>
<tr>
<td>3: perform search</td>
<td>4: return result</td>
<td>5: email result</td>
</tr>
<tr>
<td>6: provide message saying result has been sent</td>
<td>7: return error</td>
<td>8: form corrections</td>
</tr>
</tbody>
</table>
Fig. 15 Logging In Sequence Diagram

1: login details
2: check form is correct
3: if form is correct, run query
4: return error
5: form corrections
6: query result
7: create session
8: request user page results
9: user page details
10: user page

If query result is negative

11: return error
12: form corrections
Find, Link, and Assign Access Control to a Person Sequence Diagram

1. User doesn’t select anyone from the list, which causes nothing to happen.

Diagram includes:
- Linking person
- Assigning

Roger Farrow 0708955
These 3 arrays need to be linked together and incorporated into a link list rather than an array.
Fig: 20 Database Entity Relationship Diagram

ACL Codes
- ACLCode (1-31)
  - personal
  - family
  - friend
  - colleague
  - client

User
- userID
  - firstName
  - secondName
  - mainEmail (unique)
  - businessEmail (unique)
  - dob (date of birth)
  - password
  - bgColour

Person ACL
- userACL
  - userID (owner)
  - userID (recipient)
  - ACLCode

Event
- eventID
  - ACLCode
  - userID (owner)
  - eventIDate
  - eventStime
  - eventEdate
  - eventETIME
  - eventName
  - eventType
  - location
  - travelTime
  - duration
  - notes

Message
- messageId
  - userID (owner)
  - userID (recipient)
  - date
  - time
  - title
  - content

Event Person Link
- linkID
  - eventID
  - userID

Event Person Link
- userACL
  - userID (owner)
  - userID (recipient)
  - ACLCode

Roger Farrow 0708955
Fig: 21 Site Page and Function Summary

Legend:
- Blue: a distinct web page
- Green: a hidden function
- Yellow: a function of that page
- Red: an error message
- Purple: a note

Index / Log in
- Information
- Register
- Form incorrect error message
- Check Login
- If false error message

Create Session/Bean
- User's Home
- Note: Session checks are a feature of each page

User's Home
- Add Event
- Account Details
- Mailbox
- Friends
- Help Pages
- Note: Default view is today's events

Add Event
- Edit Events
  (Delete event)

Account Details
- Edit Account details
  (Delete Account)

Mailbox
- Messages
  inbox (default)
  outbox
  compose

Friends
- Search
  Sort by (Access, or name)
  Select data
  View their events
  (filtered by access control)

Help Pages

Search
- Fri ends
- email
- name

Roger Farrow 0708955
Appendix 3: The Project Poster

An Online Shareable Diary System using Sun Technologies

Student: Roger Farrow 0708955
Course: MSc Computer and Internet Applications
Supervisor: Dr. Paul Sant

Introduction

Problem
The need for an online shareable diary came from a company who were finding it difficult to track their employees during the day using a traditional paper-based diary solution. They had tested a variety of software solutions (Outlook and Call) but these solutions didn’t get well across the variety of operating systems employed by the company. So a solution was sought that was free of operating system dependency yet robust and secure and could be shared between colleagues.

Main Aim and Objectives

The Main Aim:

To produce a web-based diary solution which is shareable with colleagues that uses Sun Technologies.

Objectives:

1) Develop an online Java-based diary management system that allows users to add, view, and edit diary entries.
2) Develop a platform-independent solution that is compatible with a wide range of operating systems and web browsers.
3) Produce an ergonomic intuitive GUI with some customisation options.

Sun Technologies


MySQL 5.1: an open source database solution that is up to 90% less expensive than its competitors (Sun Microsystems Inc., p. 9, 2009).

GlassFish v2: an enterprise grade web server for serving dynamic java based websites, and has superior features to Tomcat (Sun Microsystems Inc., p. 7, 2009).

NetBeans IDE 6.7.1: a development tool with outstanding integration with GlassFish (Java net, 2009).

Sun Support: the Sun server Stack and operating systems are supported comprehensively by a single organisation which is a stark contrast to its competitors (Crimson Consulting Group, 2009).

Prototyping Methodology

The methodology was chosen for a variety of reasons:

1) The client was uncertain about their needs and how to describe them, being able to trial a version enabled them to clarify or redefine their meaning if necessary.
2) The client was concerned about uptake and usage, and this methodology helped the client and the designer work closely together to redevelop the interface considerably easier, it also acted as a check list to ensure all the components were present.

References


Design and Implementation

Design
The design process was driven by best practice, survey analysis, case study applications, and the resultant research. The project was intended to be a successful diary management system with the client’s requirements, security considerations and the technical requirements being produced employing UML, modelling techniques to map the development Class and Entity Relationship models were also employed to organise the intended development’s page structure which made developing the interface considerably easier. It also acted as a check list to ensure all the components were present.

Implementation

The implementation is HTML, CSS, JavaScript, and Java and the program development started with the HTML, before progressing with CSS, forms, and the database. These components were produced successfully and tested (Java.net, 2009) before the Java start-up was tested. The implementation included inadequate hardware and classpath related issues. The hardware related issues were caused by the system not being able to run the single server version. Java environment which was not consistent with the development tool which resulted in the main development work being done on a PC. MySQL was used to persistently store the data that the interface required. There was a problem with some web application servers when it came to testing the database connection. The program was not adequately completed, and could not perform the expected functions.

Conclusions

As was stated above the program was not functional, however the client has taken advantage of the opportunity to test the program, and has found it useful for a new product, they are currently preparing a test for some of the case study applications.

For the developer the experience of taking an idea combining it with requirements, research, and questionnaire analysis to produce design and some successfully completed testing with a working prototype at the end of the project.

However, the most significant learning point has been how important testing everything that you’re going to use in a project is.

References

