

California State University, San Bernardino

CSUSB ScholarWorks

Theses Digitization Project

John M. Pfau Library

2007

Phi Beta Delta: Implementation of a self-maintaining web site

Pallavi Pillutla

Follow this and additional works at: <https://scholarworks.lib.csusb.edu/etd-project>



Part of the [Databases and Information Systems Commons](#)

Recommended Citation

Pillutla, Pallavi, "Phi Beta Delta: Implementation of a self-maintaining web site" (2007). *Theses Digitization Project*. 3275.

<https://scholarworks.lib.csusb.edu/etd-project/3275>

This Project is brought to you for free and open access by the John M. Pfau Library at CSUSB ScholarWorks. It has been accepted for inclusion in Theses Digitization Project by an authorized administrator of CSUSB ScholarWorks. For more information, please contact scholarworks@csusb.edu.

PHI BETA DELTA:
IMPLEMENTATION OF A SELF-MAINTAINING WEB SITE

A Project
Presented to the
Faculty of
California State University,
San Bernardino

In Partial Fulfillment
of the Requirements for the Degree
Master of Science
in
Computer Science

by
Pallavi Pillutla
September 2007

PHI BETA DELTA:
IMPLEMENTATION OF A SELF-MAINTAINING WEB SITE

A Project
Presented to the
Faculty of
California State University,
San Bernardino


by
Pallavi Pillutla
September 2007

Approved by:


Dr. Josephine Mendoza, Chair,
Computer Science

8-20-07
Date


Dr. Tong Yu, Committee Member


Dr. Keith Schubert, Committee Member

© 2007 Pallavi Pillutla

ABSTRACT

The purpose of the project was to develop an easy-to-maintain web site for the Gamma Lambda Chapter of Phi Beta Delta International Honor Society here at California State University, San Bernardino, which will manage complete and up-to-date information about the mission, members, officers and all the activities of the honor society.

The web site is designed to help the Phi Beta Delta - Gamma Lambda administrator maintain the site easily and efficiently. The product has many features--a password authenticated administrator login feature that enables easy-to-edit and to-save member profiles; easy-to-download documents, awards and scholarships forms; on-line member nomination capability for faculty and staff; provides details of the events held by the honor society and executive committee information.

The application is built on a MYSQL database for data storage and management and uses Java 2 Enterprise Edition (J2EE) technologies. Java Server Pages (JSP) (version 2.0) technology is used to implement the graphical user interface (GUI) and Java Servlets (version 2.4) handle the controller layer of the core functionalities. The Apache

Tomcat application server (version 5.0) is used to host the website.

The tools used to implement the Project are Open Source software tools which are free and easy to download and install.

ACKNOWLEDGMENTS

I owe my deep gratitude to a great number of people. I would like to begin by saying that without the support and commitment of my advisor Dr. Josephine Mendoza, this project would not have been possible. I would like to especially thank Dr. Mendoza for her valuable comments which provided me with thoughtful and insightful feedback, and helped me both to sharpen my arguments and to avoid countless errors and several omissions. I am deeply indebted to Dr. Mendoza for coming to my rescue and fighting all of my battles whenever I found myself in a dilemma which was pretty often.

I am grateful to Dr. Tong Yu who was very gracious and agreed to be on my committee on a very short notice. I am also grateful to Dr. Keith Schubert for being on my committee. Dr. Yu and Dr. Schubert have provided much appreciated input, interest, and encouragement as well as invaluable guidance and advice on certain aspects of the project.

I would also like to thank Dr. Josephine Mendoza and Ms. Monica Gonzales for facilitating all the paperwork that was involved in the process.

DEDICATION

This project is dedicated to my family and friends for the support and encouragement. I am also indebted to my sons Aditya and Rohit for cooperating with me so that I could accomplish this task.

TABLE OF CONTENTS

ABSTRACT	iii
ACKNOWLEDGMENTS	v
LIST OF TABLES	ix
LIST OF FIGURES	x
CHAPTER ONE: SOFTWARE REQUIREMENTS SPECIFICATION	
1.1 Purpose of the Project	1
1.2 Limitations	1
1.3 Definition of Terms	3
CHAPTER TWO: SOFTWARE DESIGN	
2.1 Introduction	11
2.2 Preliminary Design	12
2.2.1 Requirements Collection and Analysis	12
2.2.2 Conceptual Database Design	15
2.2.3 Logical Database Design	17
2.2.4 Physical Design	18
2.2.5 The User Interface	18
2.2.6 The Tools for Construction of the Site	18
2.3 Technologies Used to Develop Phi Beta Delta Site	19
2.4 Overview of the Phi Beta Delta Application	20
2.5 The Site Administrator	21
2.5.1 Site Administrator	23
2.5.2 Database Administrator	23

2.5.3 Power User	23
2.6 End-User Interface	25
2.7 Relational Database	28
2.8 Summary	31
CHAPTER THREE: PHI BETA DELTA WEB SITE	
3.1 The Application Architecture	32
3.2 Design and Organization of the Code	33
3.3 Changing Requirements	35
3.4 Precautions	36
3.4.1 System Space	36
3.4.2 Java Database Connectivity Connections	36
3.4.3 Administrator Passwords Authentication	37
CHAPTER FOUR: SOFTWARE MAINTENANCE	
4.1 Introduction	38
4.2 Troubleshooting the Application	38
4.3 Troubleshooting SQL Code	38
4.4 Software Testing	39
CHAPTER FIVE: USERS MANUAL	
5.1 Running the Application	42
5.2 Java SDK 1.5 Standard Edition (Update 6)	42
5.3 Apache Tomcat 5.5	43
5.4 MySQL Database	44
CHAPTER SIX: CONCLUSIONS	
6.1 Summary	46

6.2 Enhancements/Improvements	46
6.3 Conclusions	47
APPENDIX A: SOFTWARE INSTALLATION	48
APPENDIX B: SAMPLE SOURCE CODE	50
REFERENCES	57

LIST OF TABLES

Table 1. Minimum System Requirements for Host Machine	11
Table 2. Phi Beta Delta Application Functionalities	22
Table 3. List of Forms	25
Table 4. List of Site Links	26
Table 5. Basic Building Blocks of a Data Model	29

LIST OF FIGURES

Figure 1.	Application Model	2
Figure 2.	Conceptual Data Model: Extended Entity-Relationship Diagram - Entities and Relationships	15
Figure 3.	Conceptual Model: Extended Entity-Relationship Diagram - Entities and Attributes	17
Figure 4.	Deployment Diagram	21
Figure 5.	Use Case Diagram	24
Figure 6.	Snap Shot - Phi Beta Delta Gamma Lambda Home Page	28
Figure 7.	The Phi Beta Delta Database	30
Figure 8.	Java Server Pages Model 2 Architecture	34
Figure 9.	Phi Beta Delta - Package Diagram	35
Figure 10.	Java Environment Variable	43
Figure 11.	Apache Tomcat Windows Service	44
Figure 12.	MySQL Windows Service	45

CHAPTER ONE

SOFTWARE REQUIREMENTS SPECIFICATION

1.1 Purpose of the Project

The purpose of the project was to develop an easy to maintain web site for the Gamma Lambda Chapter of the Phi Beta Delta International Honor Society that will be an improvement over the existing web site.

The site will display information regarding the honor society for the current year. This application is designed to let the society's secretary update the database and make the necessary changes; track the active members and the details of the activities of the honor society.

1.2 Limitations

The existing site does not have many features desirable for a functional and easy-to-use web site. The existing tool is not very efficient, nor is it very secure. Scanning for some information on a web site can be monotonous, boring, error prone, and time consuming. The administrator of the web site needs to provide data to the end-user that is deemed important and useful. Precise information with needful insights has to be provided by the administrator. The architecture of the web site proposed here will help in such situations.

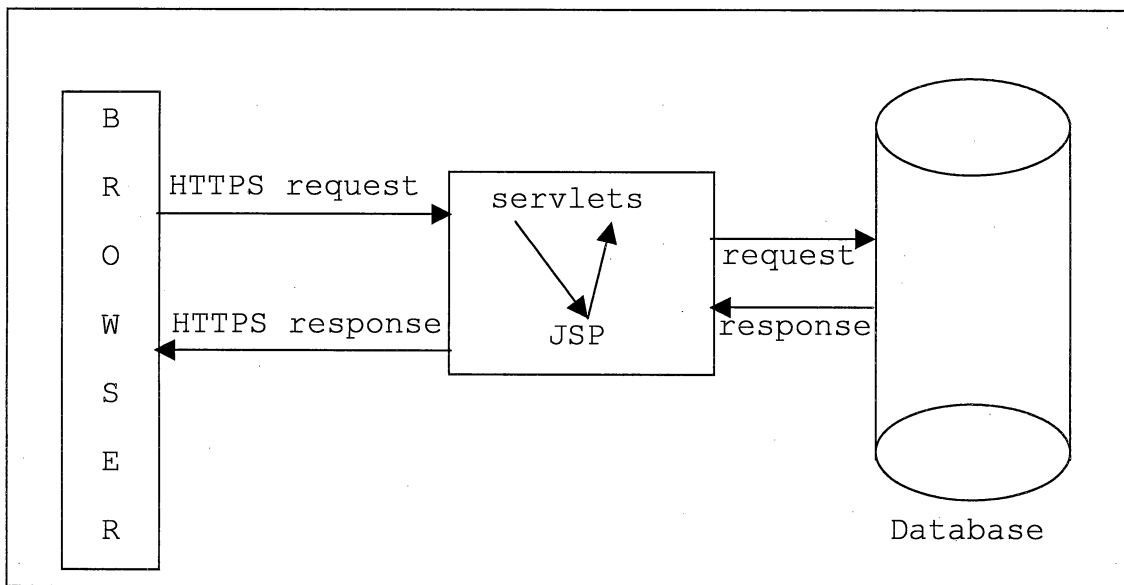


Figure 1. Application Model

During the development of the project, a number of limitations for the existing Phi Beta Delta web site were noted. These limitations are:

1. The site did not have a stable database. The data was not up-to-date. The tables were not normalized causing inconsistency in data. There was no member login and password authentication on the database side.
2. The user interface was not friendly. It was tedious to navigate through the site. Important information was not presented efficiently to the user. Easy to use and downloadable forms in word or PDF document format did not exist.

3. The inefficient application architecture affected performance. It caused unnecessary time lags and led to unnecessary downtime.

1.3 Definition of Terms

The following terms are defined as they apply to the project and taken from books by Dietel et al. [1] and Schmitt [11].

Apache Tomcat: Apache Tomcat is a Servlets engine and implements the Java Servlets and Java Server Pages specification. Apache Tomcat is developed in an open source environment and is one of the mostly widely used application servers.

Apache Tomcat is supported on several operating systems including Windows and most versions of UNIX.

API: API is the acronym for application program interface. API is the specific method prescribed by a computer operating system or by an application program by which a programmer writing an application program can make requests of the operating system or another application. It is also the methods prescribed by libraries of functions and classes. An API can be contrasted with a graphical user interface or a command interface (both of which are direct user

interfaces) as interfaces to an operating system or a program [3].

Client/Server: Client/Server architecture involves client processes requesting service from server processes.

In this architecture one program, the client makes a service request to another program, the server, which fulfills the request. For Internet applications, the web browser is a client program that requests services (sending of web pages or files) from a web server in another computer on the Internet [1].

CSS: CSS is an acronym for Cascading Style Sheets. Style sheets offer much more flexibility in terms of the presentation effects of a web page. A style sheet is a definition of how content should be rendered on the web page. Properties such as color, background, margin, border, etc can be applied to make the web page attractive and interesting. A style sheet separates the content from style [11].

GUI: GUI is an acronym for Graphical User interface. GUI interface allows users to navigate and interact with applications with simple mouse clicks [11].

A good GUI enables users to interact with applications easily. Intuitive and easy to use user

interfaces are essential for all applications that have a user interface.

HTML: Hypertext Markup Language (HTML) is used for the creation of web pages and web content. HTML separates the web content from formatting. It consists of set of tags and attributes to create web pages [11].

HTTPS: HTTPS is a URI (Uniform Resource Identifier) scheme which is equivalent and syntactically identical to the HTTP scheme normally used for accessing resources using HTTP. Using https URL indicates that HTTP is to be used, but with additional security measures applied to the transactions.

HTTPS is not strictly, a separate protocol, as the data is still transferred using HTTP; however, instead of using plain text socket communication, the session data is encrypted using a version of the Secure Socket Layer (SSL) or Transport Layer Security (TLS) protocols ensuring security [12].

IP: IP is an acronym for internet Protocol and is used to transfer information in a network. IP is a lower level network layer protocol in the internet suite of protocols. IP addresses are globally unique, 32-bit numbers assigned by the Network Information Center.

IP is responsible for moving packet of data from node to node [1].

Java Bean: Java beans are re-usable software components written in the Java Programming language. In order for a piece of code written in Java to be considered a Java bean, it has to conform to the specification of the Java programming language with respect to their construction and behavior. By virtue of conforming to the specification, Java beans state can be persisted in a file system or a relational data base.

Java EE: Java EE stands for Java platform Enterprise Edition. It is a programming platform. The Java EE platform is defined by a specification. It includes several API specifications, a set of services and protocols that provide the functionality for developing multi-tiered, web-based applications. It is designed to simplify complex problems with the development, deployment and management of multi-tier enterprise solutions. Java EE is a standard not a product. The Java EE architecture is based on the Java programming language. It enables developers to write code in Java. On the client side, Java EE supports HTML and Java applets or applications. It

relies on Java Server Pages (JSPs) and servlet code to create HTML or other formatted data for the code [3].

JDBC: JDBC stands for Java Database Connectivity. JDBC technology is an API that provides database connectivity to a wide range of SQL databases and access to other data sources. It is the API which interfaces with the database. JDBC enables SQL statements to be embedded inside Java API calls. In a Java application, using standard library routines a connection to the database is established and then using JDBC SQL, code is sent to the database and the results returned are processed. The connection is closed once the transaction is done. The set of Java classes that implement the JDBC interfaces for a particular database engine is called a JDBC driver. JDBC provides maximum flexibility to the developers to access the data from the database [8].

JSP: JSP is the acronym for Java Server Pages. JSP technology provides a simplified and fast way to create web content. JSP technology enables rapid development of web-based applications that are server and platform-independent. It separates the user interface from content generation, enabling designers

to change the overall web page layout without altering the underlying dynamic content of the page. JSP incorporates simple tag handlers, which utilize a new and much simpler and cleaner tag extension API. This generates pluggable, available re-usable tag libraries which reduce the amount of code needed to write powerful web applications [5].

JSP technology uses tags that encapsulate the logic that generates the content for the page. The application logic can reside in server-based resources that the page accesses with these tags. Any and all formatting (such as HTML) tags are passed directly back to the response page. By separating the page logic from its design and display and supporting a reusable component-based design, JSP technology makes it faster and easier to build web-based applications [4].

Servlets: Servlets are platform-independent, server-side modules that fit seamlessly into a web server framework and can be used to extend the capabilities of a web server with minimal overhead, maintenance, and support. Unlike other scripting languages, Servlets involve no platform-specific considerations or modifications. Servlets are application components

that are downloaded, on demand, to the part of the system that needs them. Servlets and JSP technology together provide dynamic web programming since they are platform-independent, separate logic and content from display, easy to administer and easy to use [9].

TCP: TCP is an acronym for Transmission Control Protocol. TCP is the transport layer protocol in the five layer TCP/IP model. TCP is a connection-oriented transport protocol that sends data as an unstructured stream of bytes. By using sequence numbers and acknowledgment messages, TCP can provide a sending node with delivery information about packets transmitted to a destination node [1].

Application layer protocols such as HTTP, FTP etc. use the TCP/IP model for transferring data.

UML: UML is the acronym for Unified Modeling Language. It is a standard language for specifying, presenting and documenting the components of software systems. The UML represents a graphical notation of software systems. It is especially useful in developing object-oriented software. UML employs techniques to manage the complexity of large systems and automate production of software. In addition to providing expressive visual modeling language, UML also

provides extensibility and specialization mechanisms to extend the core concepts such as collaboration, framework, patterns and components. UML is independent of programming languages and development process. UML can also be used to represent the database schemas. Objects of a database such as tables and views can be represented using UML diagrams. UML modeling tool can be used to graphically reflect relationships between these objects [2].

CHAPTER TWO
SOFTWARE DESIGN

2.1 Introduction

The purpose of building the Phi Beta Delta web site is to provide the user an easy and instant access to information. The Phi Beta Delta web site is also designed such that it is easy to maintain as well. The site has features such as quick-to-load web pages with meaningful content. The content on the web pages is useful, appropriate and up-to-date. It is arranged logically with the most current information displayed first. Striking a balance between the content and look and feel of the web page is important for any site. This has been achieved by using the CSS and Java tools.

Table 1. Minimum System Requirements for Host Machine

Processor Type	Pentium 4
Processors Speed	300 MHz
System memory	512 MB
Hard Disk	25 MB of free hard disk space for installation
Operating System	Microsoft Windows XP

2.2 Preliminary Design

The methodology to construct the Phi Beta Delta web site consisted of the following steps:

2.2.1 Requirements Collection and Analysis

In this phase, the database requirements were obtained and the key user scenarios were identified.

The requirements for building the Gamma Lambda Chapter of the Phi Beta Delta Honor Society have been primarily gathered from meetings and discussions with my faculty advisor, Dr. Josephine Mendoza. The primary requirement of the new site is that it be easy to maintain with minimal human intervention.

Based on these discussions the following functional requirements for the web site have been identified.

a. Authentication

A security feature to authenticate valid users of the site will be implemented. Proper authentication of the users will place restrictions on the end-users accessibility rights to the system. The administrator will be given the rights of a 'super' user. The administrator has the option to grant privileges to different end-users. Different types of end-users will have different levels of access within the system. These users include general users who have

the ability to browse the information, site administrators who have the ability to edit content, and power users who have the ability to submit nominations on line.

b. User-Friendly Website

A user-friendly interface will be provided to the end-user to interact with the system. The interface will clearly display the contents of the site and provide appropriate information to the user. The administrator will be provided with sufficient flexibility to edit the contents of the site through a user-friendly interface.

c. Member Profiles

The profiles of the society members--name, position, phone number and email--will be displayed. The listing displayed will always be current. Individual profiles to be shown include those for executive board members and officers, faculty members, staff members and student members. The administrator of the site will update these listings as the changes take place. The administrator of the site will have the ability to 'create', 'edit' and 'delete' profiles of members.

d. Activities Listing

The site will have a calendar of recent as well as up-coming activities. This will keep the users and members informed about the various events. Examples include upcoming events, recent events and announcements with links to the details of the events.

e. Applications/Forms

The site will allow the download of various forms in electronic form. Such forms include membership, nomination, letters of recommendation and other documents such as the Phi Beta Delta Chapter manual.

f. Evaluation

An evaluation component is one of the functional requirements. The site will provide the appropriate committees a place to evaluate scholarship and membership applications.

g. Recruiting

The site will enable the members to nominate new and honorary members to the society in accordance to the guidelines of the Phi Beta Delta Chapter.

h. Database component

The database will store details of the faculty, staff and student members and provide an easy means to

update the web site with the latest information without extensive code changes.

2.2.2 Conceptual Database Design

The conceptual database design step involved reviewing and understanding the database requirements to support the functional requirements identified in the requirements phase. Entities and their attributes were identified; relationships among these entities were analyzed. These were then represented graphically using the Entity-Relationship (ER) diagrams.

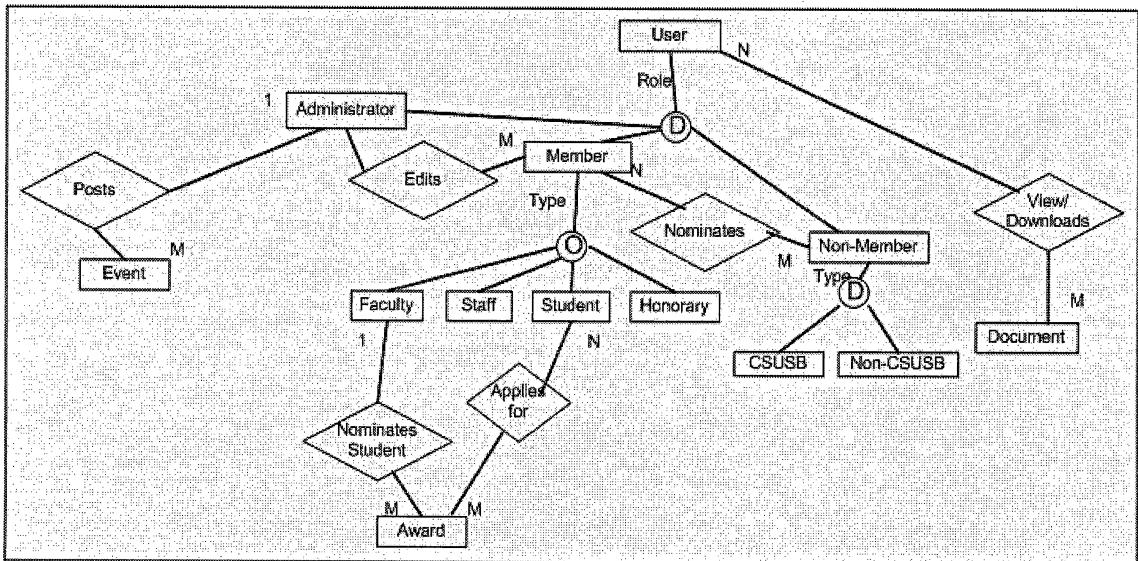


Figure 2. Conceptual Data Model: Extended

Entity-Relationship Diagram - Entities and Relationships

In Figure 2, entities are represented as rectangles; relationships are indicated by diamonds; attributes of the

relationships (if any) are depicted as ovals. In Figure 2, the entity "User" is a super class of disjoint subclasses Administrator, Member, and Non-Member. This indicates that there are three user types in the system - Administrator, Member and Non-Member. The "D" inside the circle indicates disjoint participation. The Member subclass in turn has overlapping subclasses - Faculty, Staff, Student and Honorary. This describes that a member can be either a faculty, staff, student or honorary member. The overlapping membership indicated as "O" inside the circle means that a member can be both a staff and a student. The cardinality of the relationships are indicated by the symbols, 1-M (one-to-many) and M-N (many-to-many).

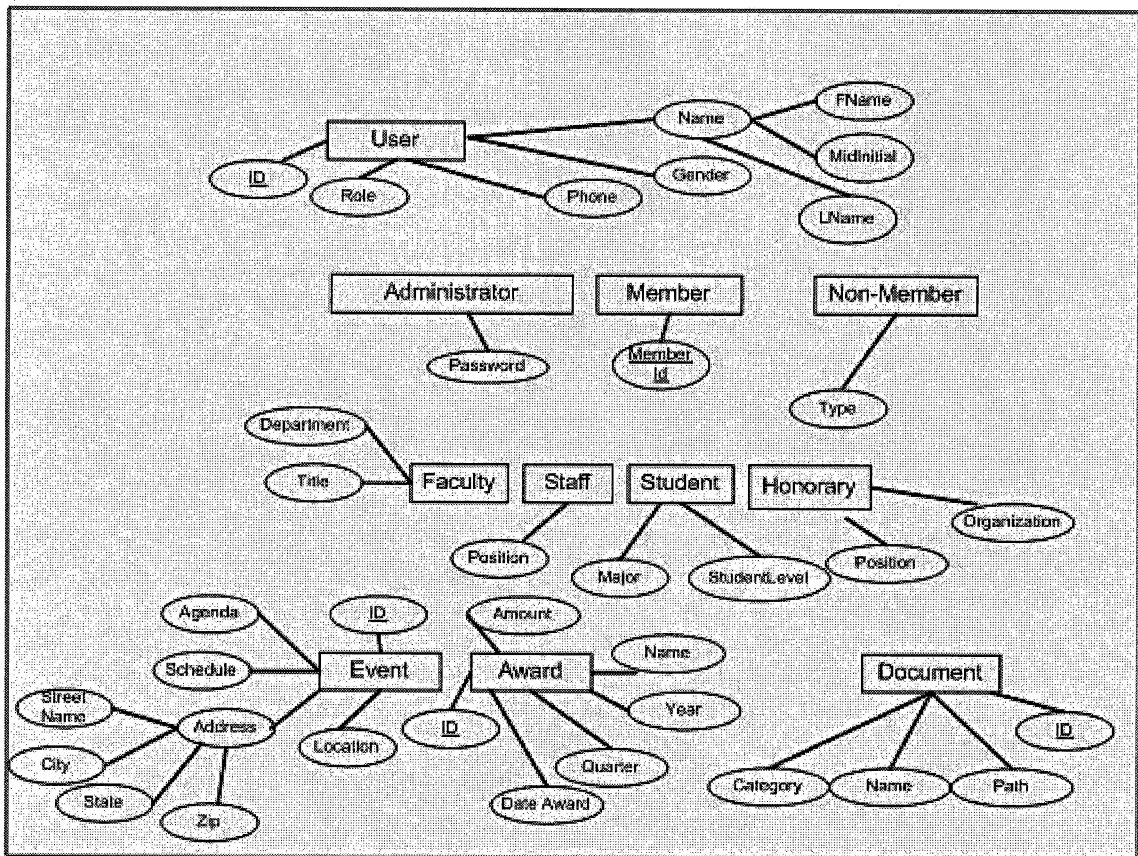


Figure 3. Conceptual Model: Extended Entity-Relationship Diagram - Entities and Attributes

In Figure 3, composite attributes such as Address are indicated by having its component attributes listed using ovals. Key attributes of an entity are underlined.

2.2.3 Logical Database Design

Logical database design involves conversion of the ER model to the relational data model and normalization of the relational data model. The normalization process ensures that insertion, deletion and update anomalies never occur.

2.2.4 Physical Design

The physical database design step involved selecting file formats and access paths and creating relational tables in MYSQL. In addition data loading and creating query scripts to access and manipulate the MYSQL database were also performed.

2.2.5 The User Interface

The design of the user interface was based on the end-user requirements. During the design phase, the links to different web pages to navigate the site for information were identified. These include links for information about Phi Beta Delta, Officers, Society Members, Honorary Members, Events, Awards & Scholarships, Forms, Nominations, Administrator access and a link with information about contact information.

2.2.6 The Tools for Construction of the Site

In the next step the tools that can be used to build the site were identified. The primary consideration for selecting the tools was to minimize costs. Open source standards, technologies and tools were selected in the implementation of this application. These include Java programming language, Java 2 Enterprise Edition technologies such as Servlets and Java Server Pages, MySQL database and Tomcat Servlets engine.

2.3 Technologies Used to Develop Phi Beta Delta Site

The following is a description of the software tools and languages that were used to develop the Phi Beta Delta Web site. These tools and technologies are widely used and conform to standard specifications -- Java programming language, Java 2 Enterprise Edition technologies such as Servlets and Java Server Pages and MySQL database management system. Following sections describe in greater detail the various technologies.

Java and J2EE: The Phi Beta Delta web site was developed using Java programming language and Java 2 Enterprise Edition (J2EE) technologies. Java programming language allows the application to be deployed in any operating system without re-writing the code. Sun Microsystems provides a set of APIs called the Java Development Kit (JDK) to enable application development in Java. JDK is easy to download from the Internet free of cost. JDK is not only easily configurable but it also has libraries and frameworks that support web development. The web components are either servlets or pages created using JSP technology. Servlets are Java programming language classes that dynamically process requests and construct responses. JSP pages are text-based documents that execute

as servlets but allow a more natural approach to creating static content.

MySQL Database Management System: MySQL database is used for the back-end storage of the data. MySQL database is the most widely used open source database. It is well-known for its consistent fast performance, reliability and ease of use. It is also supported on a variety of platforms including Linux and Windows, among others. MySQL supports high performance ACID (Atomicity, Consistency, Isolation, Durability) compliant transaction support. Such support enables it to be used in e-commerce and other mission-critical applications [16].

Apache Tomcat Server: Apache Tomcat is a Servlets engine and supports the Java Servlets and Java Server Pages specification of the Java 2 Enterprise Edition. It is developed as an open source tool and powers numerous web applications across various industries and organizations. For the current project, Apache Tomcat is used to host the Phi Beta Delta website.

2.4 Overview of the Phi Beta Delta Application

The Phi Beta Delta web site provides information to a general user of the application without having the need to login. Site administrators who will have the privileges to

edit content and power users who will have limited ability to edit content, such as submitting online nominations will be required to log in with valid credentials.

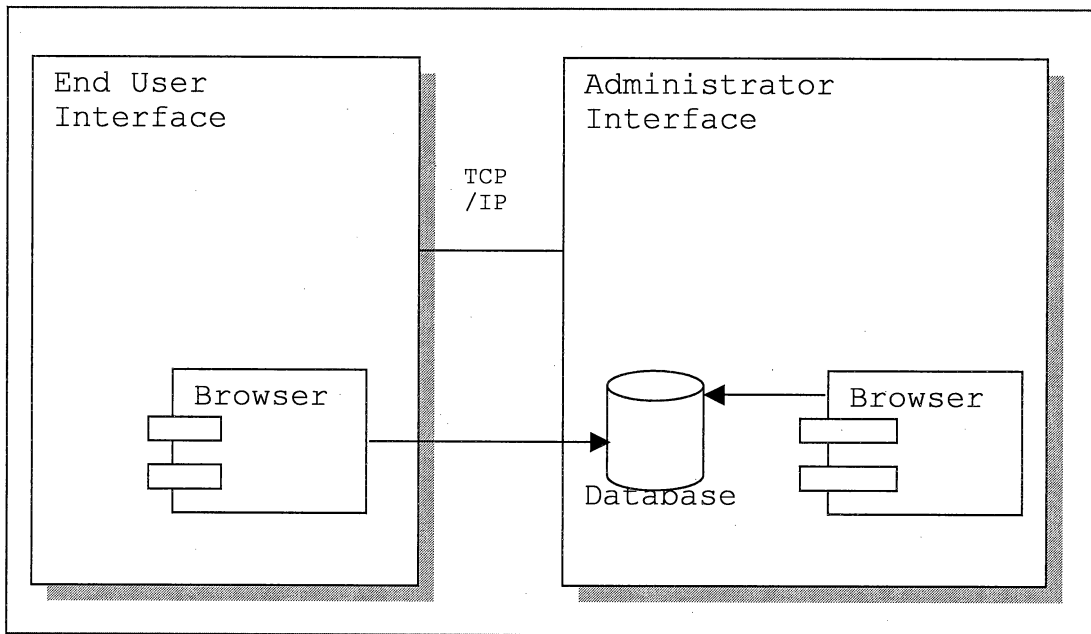


Figure 4. Deployment Diagram

2.5 The Site Administrator

The site administrator interface sets up and manages all aspects of the application interface as well as the functionalities. The database administrator manages the database aspects. All the data is stored in the tables of the database. The application is built on MYSQL database which stores all the information.

Table 2. Phi Beta Delta Application Functionalities

2.1	Display information about Phi Beta Delta Honor Society
2.2	Display information about all members of the honor society
2.3	Display information about officers of the honor society
2.4	Display details of the events sponsored by the honor society
2.5	Display details of the awards and scholarships given to the members
2.6	Display various forms available
2.7	Make the forms downloadable
2.8	Display honorary members of the honor society
2.9	Authenticate the password in the administrator log-in
2.10	Edit member profiles by the administrator
2.11	Update events by the administrator
2.12	Update awards and scholarships by the administrator
2.13	Nominate members online by members of the honor

The application supports three levels of access. Any end-user can access the site and browse the site for general information. Additionally, there is a power user log-in, a site administrator log-in to the application. Database administrators are also required to have valid

credentials to log-in to the database. The site administrator has full access to all the different modules of the application. The three primary types of user log-ins are:

2.5.1 Site Administrator

The site administrator has the final control and responsibility for the entire application. Only the site administrator has full access to the application. The site administrator has the ability to make changes to the web pages by adding or deleting data from the user interface.

2.5.2 Database Administrator

The database administrator can add tables to the database and also modify the data in the tables directly. The database administrator can also add new user profiles and create password authentication for users at different levels.

2.5.3 Power User

Power users are granted limited access to edit the site. These users have the capability to submit nominations online. The nomination details will be stored in database tables.

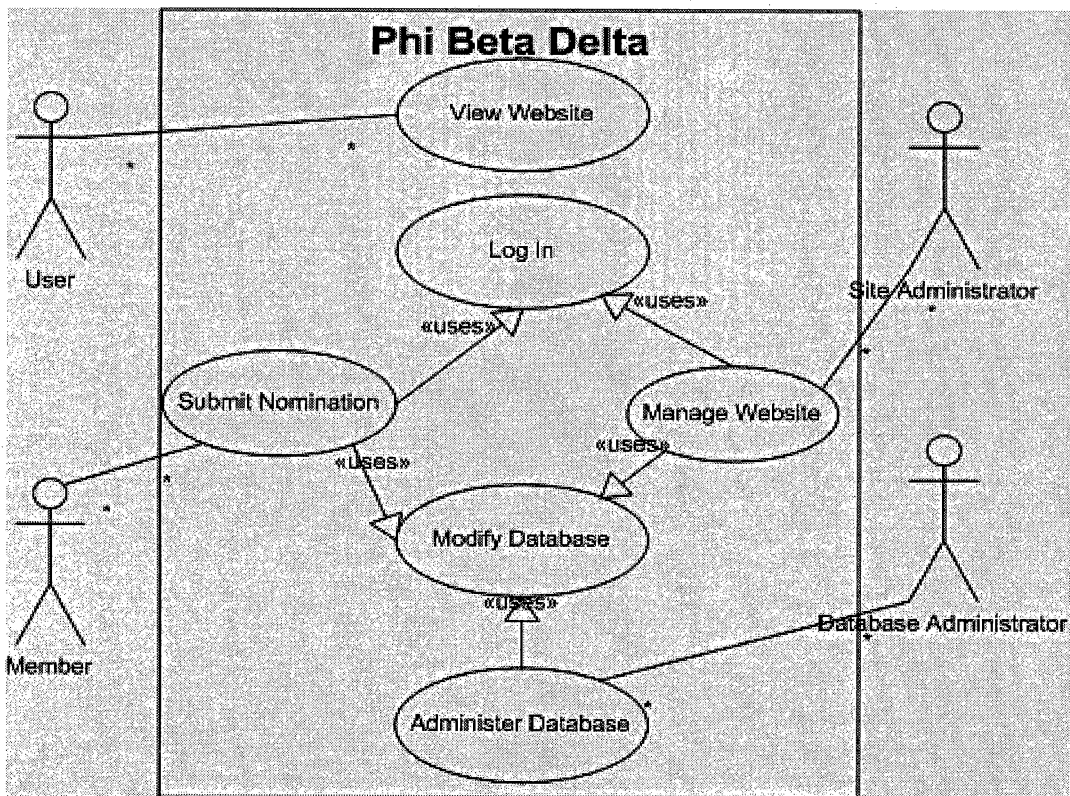


Figure 5. Use Case Diagram

The Phi Beta Delta application includes a number of forms that are easy to download in the word or PDF document format. These documents are made accessible to the users in order to make it convenient for the users to use the web site. The user can immediately print the required form and send it to the Phi Beta Delta office without any delay. Following is the list of forms that the user can conveniently print directly from the site.

Table 3. List of Forms

Membership Forms	Phi Beta Delta Membership Renewal Form
	Phi Beta Delta Faculty/Staff New Member Form
	Phi Beta Delta New Student Member Form
Nomination Forms	Phi Beta Delta Faculty/Staff Nomination Form
	Phi Beta Delta Student Nomination Form
	Phi Beta Delta Awards Nomination Form
	Phi Beta Delta Scholarship Nomination Form
Recommendation Forms	Phi Beta Delta Student Recommendation Form
Other	Phi Beta Delta Chapter Manual

2.6 End-User Interface

Navigation is an integral part of a web site. Navigation capability in a web site not only gives the end-user the choice, convenience and flexibility to access the information but it also generates user's interest in the web site. The Phi Beta Delta site has simple to understand, informative, well-established and functional links.

The end-user can only view the content of the web page. The end-user will not be able to update either the content or the display of the interface. The end-user will be provided with complete and detailed information about the Phi Beta Delta Society and its activities. The end-user can access this information making use of easy-to-navigate links. The names of the links are easy to understand and self-explanatory. The end-user is provided with the following links

Table 4. List of Site Links

Link	Description
About Phi Beta Delta	Describes the history of the Phi Beta Delta organization and its mandate.
Officers	Presents information about the board members and the executive committee for the current year. When a user is logged in as an administrator this link takes the user from a read-only mode to edit mode. In the edit mode, the site administrator can add new officers, delete existing officers and edit information about existing officers.
Society Members	Presents information about existing society members. The information presented as a table can be sorted by various columns as well. When logged-in as a site administrator, a user has the option to edit the information about existing members, add and/or delete members.

Link	Description
Honorary Members	Presents the users with information relating to the honorary members of Phi Beta Delta. Honorary Members are those that the society desires to honor in a special way and who would otherwise not qualify for regular membership. In an edit mode, the site administrator can add new members and modify or delete existing honorary members.
Awards & Scholarships	Presents the user with information about the recipients of awards and scholarships. When logged in as an administrator a user can add, modify and delete information about the scholarship recipients.
Forms	Allows users of the website to download various documents in a word or PDF document format. These include membership forms, nomination forms, and letter of recommendation forms.
Nominations	Enables members to nominate a student, staff or faculty for membership to the society. When logged in with this role, users can add new nominations and update existing nominations.
Administrator	Presents a log-in screen. When logged in with administrator credentials the users will be able to edit information in any link.
Contact Us	Presents users with general contact information for the Gamma Lambda Chapter of the Phi Beta Delta web site. This includes address and phone numbers for the chapter.

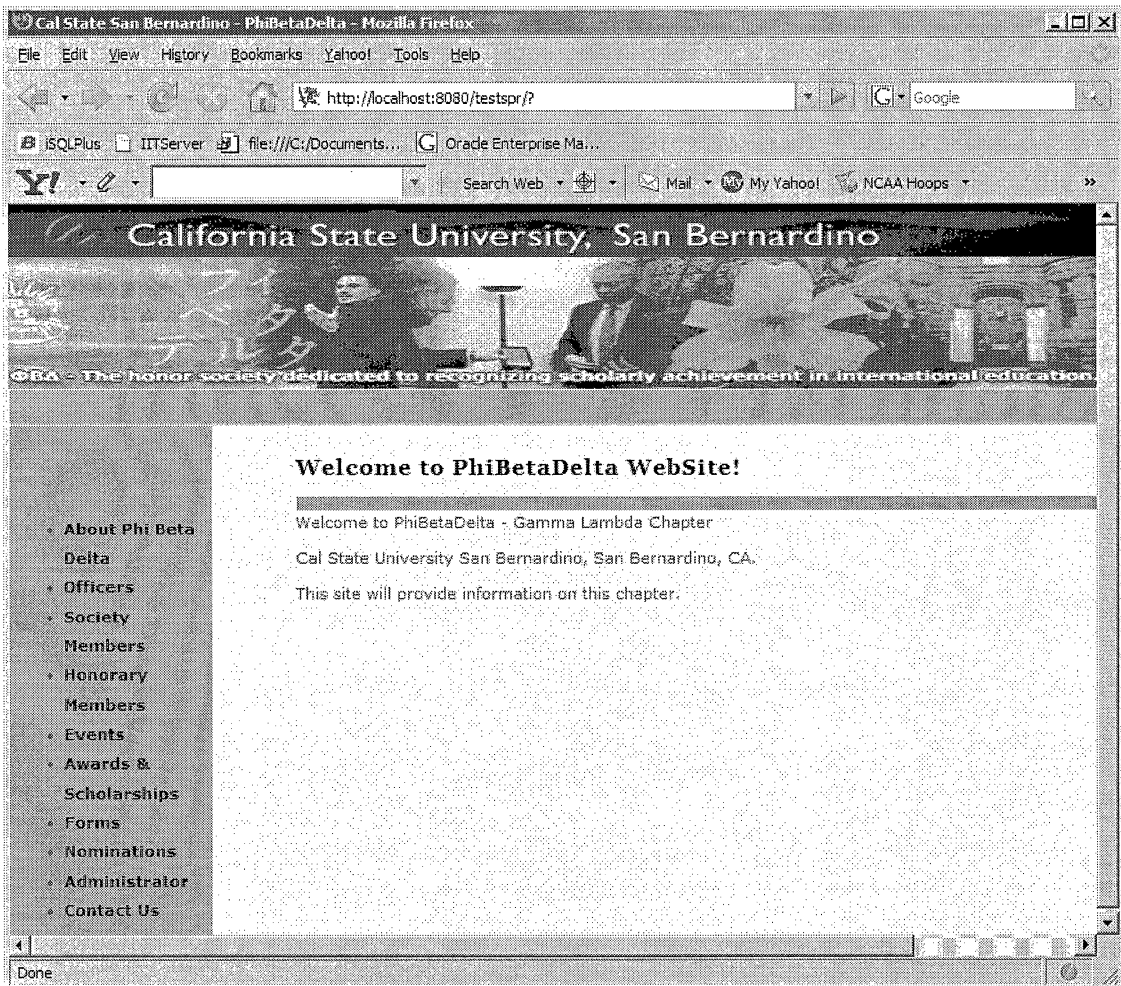


Figure 6. Snap Shot - Phi Beta Delta Gamma Lambda Home Page

2.7 Relational Database

The main objective of data modeling is to arrange the smallest information (attributes) of the data model into logical sets so the storage and retrieval is efficient and anomalies are removed. In a data model the basic building blocks are:

Table 5. Basic Building Blocks of a Data Model

Columns/attributes:	Descriptors that characterize the entity.
Referential Integrity:	Relationship(s) mapped as column(s).
Tables:	Logically related set of information.

2.8 Summary

The Phi Beta Delta application presents information about the Phi Beta Delta chapter to the users through various easy to navigate links. Users can access information without having to log in to the web site. Site administrators with access privileges can log in to the web site to edit information that will be presented through the web site. The application has been developed using open source tools and technologies such as Java programming language, Java 2 Enterprise Edition technologies (J2EE) and MySQL relational database. Apache Tomcat Servlets engine powers the application as a web server and Servlets container.

CHAPTER THREE

PHI BETA DELTA WEB SITE

3.1 The Application Architecture

The design of a software system not only affects the performance of the application but also its modifiability, reliability, security and usability. It is important to understand the kind of modifications the application will support at the design time itself.

A web site should be easy to maintain and extend. Clear separation has to be maintained between the presentation, functional and database tiers of the web application. These features have been incorporated while building the Phi Beta Delta web site by using a variety of open-source software. They include Apache Tomcat Servlets Engine and MySQL database. The architecture for the site is designed to have optimum performance at the same time keeping it user-friendly and easy to use. The architecture of the software determines the design flexibility of a solution. This has been incorporated into the Phi Beta Delta website by using Model-View-Controller architecture to separate out the presentation web pages from the business logic. The links to the web-pages help navigate to the appropriate information with minimal number of

clicks. MySQL database is a proven high performance open source database and has been used in thousands of installations all over the world. The high performance of the database coupled with the reliability of the database makes it very valuable infrastructure software to the Phi Beta Delta web application. In addition, the licensing costs of buying commercial database software are eliminated by using a proven open source solution. Similarly the Apache Tomcat Servlets engine is an open source software with proven track record in reliability and adherence to the Java Enterprise standards. It is the software behind several mission critical large-scale applications across numerous companies and industries [14]. Examples of Apache Tomcat deployments include in companies such as E*Trade Financial, Wal-Mart, EMC/Documentum and The Weather Channel [15].

3.2 Design and Organization of the Code

The Phi Beta Delta application has been developed as a 3-tier application. It uses the Model 2 design approach [13]. Model 2 design is an extension of the popular MVC design pattern and used in J2EE applications utilizing Servlets and Java Server Pages.

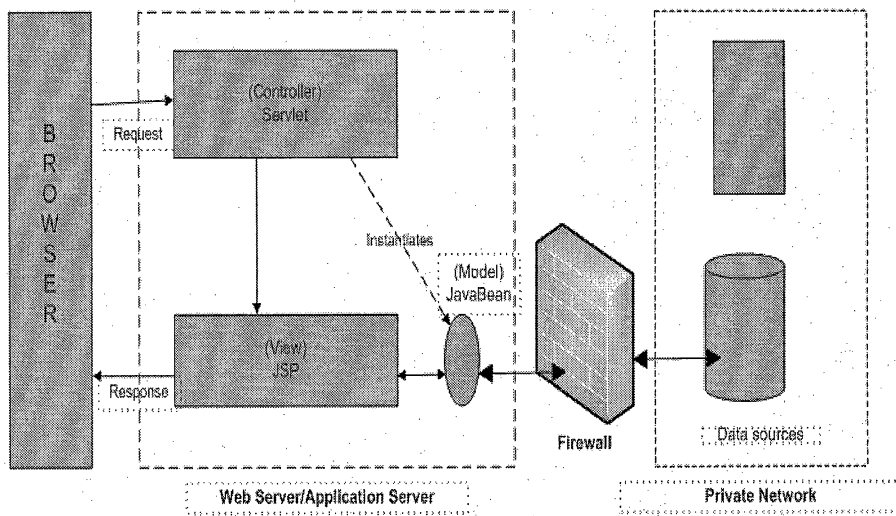


Figure 8. Java Server Pages Model 2 Architecture

In the Model 2 Architecture as shown in Figure 7 above, Servlets act as controllers. To get the relevant data from the database, java beans are used to retrieve or update the relevant information from the database. After the desired information is collected the information is passed to the Java Server Pages which present the information to the user. This design approach is highly flexible and separates content from presentation.

The code for Phi Beta Delta is organized into three primary packages. These include web, model and data access packages. The web package holds all the Servlets that act as controllers. The data access package has the code necessary to establish database connections and retrieve or update database information. The model package has the java beans that aid in the transfer of information from

the controller layer to the data access layer and from the data access layer to the presentation layer, which includes the Java Server Pages that present information to the end-user.

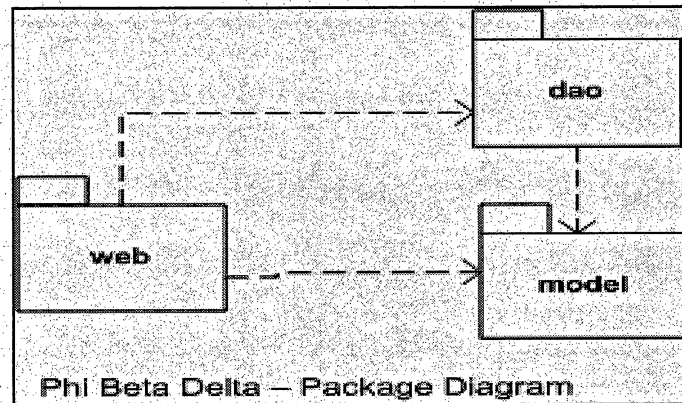


Figure 9. Phi Beta Delta - Package Diagram

3.3 Changing Requirements

All application development is characterized by constant evolution because of the frequent and sudden changes of the core requirements of a software project. The challenge for software development is to devise a system architecture that allows continuous, incremental updates while meeting the requirement that the application can react to changes in its environment. As there are various possible changes that can be made to the application, the design of the application must ensure extensibility.

Application failures and sudden, slow performance are usually the result of changes to the application. Developers planning on implementing new changes should have the knowledge of application design and program code which is vital in reducing downtime. To maintain performance the database schema should also be kept consistent. Random changes to the database will result in disorganized data affecting application performance.

3.4 Precautions

There are some precautions that should be taken while installing and running this application:

3.4.1 System Space

The dimensions of the system space required by the application must be specified accurately. The application and the supporting software require approximately 400 MB of free disk space.

3.4.2 Java Database Connectivity Connections

Database connections to access the database from the application are configured in the configuration files of the Apache Tomcat Servlets engine which hosts the applications. Configuring these values outside the application code allows the flexibility to change the

configuration during deployment without modifying the code.

3.4.3 Administrator Passwords Authentication

To protect passwords and prevent unauthorized access to the application, it is a good practice to change passwords periodically such as once every month. The user typed passwords are authenticated against the database to validate the credentials. Tomcat's database realm is used in the application for password authentication.

CHAPTER FOUR

SOFTWARE MAINTENANCE

4.1 Introduction

The maintenance steps may involve modifying the java code, the SQL queries as requirements change slightly or problems are found during the use of the web site. Other aspects of maintenance include deploying the application, functional and performance testing. The following sections describe the various steps in greater detail.

4.2 Troubleshooting the Application

Tomcat produces a few log files when the application is running. These log files are located under the 'logs' directory under the Tomcat installation directory. The log files provide such information as what clients from which machines have accessed the application, more information on errors encountered during the application and other information.

4.3 Troubleshooting SQL Code

SQL errors encountered during the application execution phases also get logged to the Tomcat log files. The exceptions captured in the log files point to the specific lines of code that has caused the errors. These

can be diagnosed by any person having access to the source code of the application.

4.4 Software Testing

The data entered through GUI interface was verified to check if the data was stored in the correct tables and columns in the database and that the constraints were observed. The data entry fields were tested for out-of-range data (both length and value). Also verified that all the data entry fields in the application accepted valid data and rejected invalid data. The application was tested to check if the data was correctly redisplayed when re-visited.

Functional testing was also done to make sure that each of the operations functioned the way it was supposed to be done. Functional testing of the Phi Beta Delta web application can be performed by clicking on the various links of the application. These include links about members, events, awards, viewing and downloading forms. In addition, functionality available to power users such as submitting nominations can be tested. Furthermore, a user logged into the application as an administrator of the application, can test the functionality available to administrators. Such functionality includes editing member

information, adding and deleting members and editing events.

Java programming language provides a solid mechanism to trap run-time errors during the execution of the application. The Phi Beta application source code uses try-catch blocks to catch run time errors as well as user errors. User errors include such examples as when the user does not enter all the required information in a form. In such instances errors are trapped and the user is provided with a friendly message to fill the required information. In cases when unusual and abnormal errors occur during the run time of the application, user-friendly messages are presented so user can contact a system administrator to help resolve problems.

Performance of any data operation depends on a number of factors. Data access techniques that use stored procedures reduce the load on the database as compared to approaches that use dynamic SQL expressions. The number of round trips made to the database server is also a factor, especially where locks and transactions span multiple round trips. The amount of data sent over the network is another important factor. It depends upon the number of SQL Statements. It depends on the database activity. It also depends on the hardware configuration of the user's

machine. There are several variables that influence performance.

CHAPTER FIVE

USERS MANUAL

The following sections describe the various steps involved in running the Phi Beta Delta website.

5.1 Running the Application

The Phi Beta Delta application is deployed in Tomcat Servlets Engine which serves to host the application. The back-end database is a MySQL database. Both of these software run as windows services and start automatically when the machine is started.

5.2 Java SDK 1.5 Standard Edition (Update 6)

The Java SDK provides the Java Virtual Machine used by both Apache Tomcat and the Phi Beta Delta application. It is invoked by the application server when the Phi Beta application is started.

In order to find the Java SDK, the `JAVA_HOME` environment variable must be set. To check (or change) the value of this variable, bring up the "System" tool located in "Control Panel" and go to the "Advanced" tab. Press the "Environment Variables" button at the bottom of the page and look for the `JAVA_HOME` environment variable. It should appear as follows:

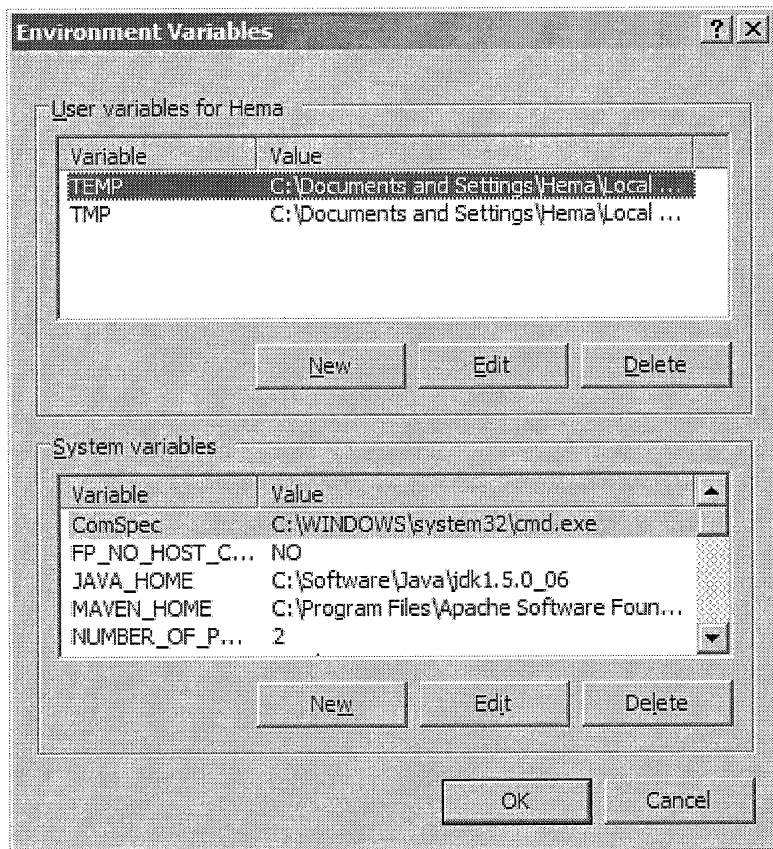


Figure 10. Java Environment Variable

5.3 Apache Tomcat 5.5

Apache Tomcat is the application server on which the Phi Beta Delta application runs. It is implemented in the form of a Windows service and is responsible for presenting the web interface to the user. It is also the container in which Phi Beta Delta Application runs. To stop or start the Tomcat service, bring up the "Services" management tool in the "Administrative Tools" program group. The following screen will appear which will provide

the ability to check the status of the service and if necessary to stop or restart the service.

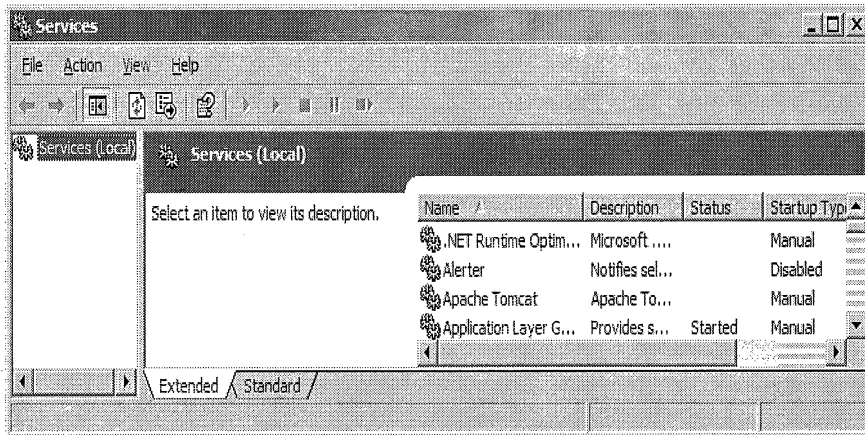


Figure 11. Apache Tomcat Windows Service

There are numerous configuration files associated with Apache Tomcat and it would be impossible to cover them all in this document, but perhaps the single most important configuration file would be "server.xml". This file specifies the main port on which the application server will listen for requests (port 8080). The file is located in the following directory. However, it should not be necessary to edit this file under normal circumstance.

F:\Tomcat 5.5\conf\server.xml

5.4 MySQL Database

MySQL is the back end relational database used by the application. All the data of the application is stored in

the database. MySQL is installed as a windows service and can be started and stopped by starting and stopping the service.

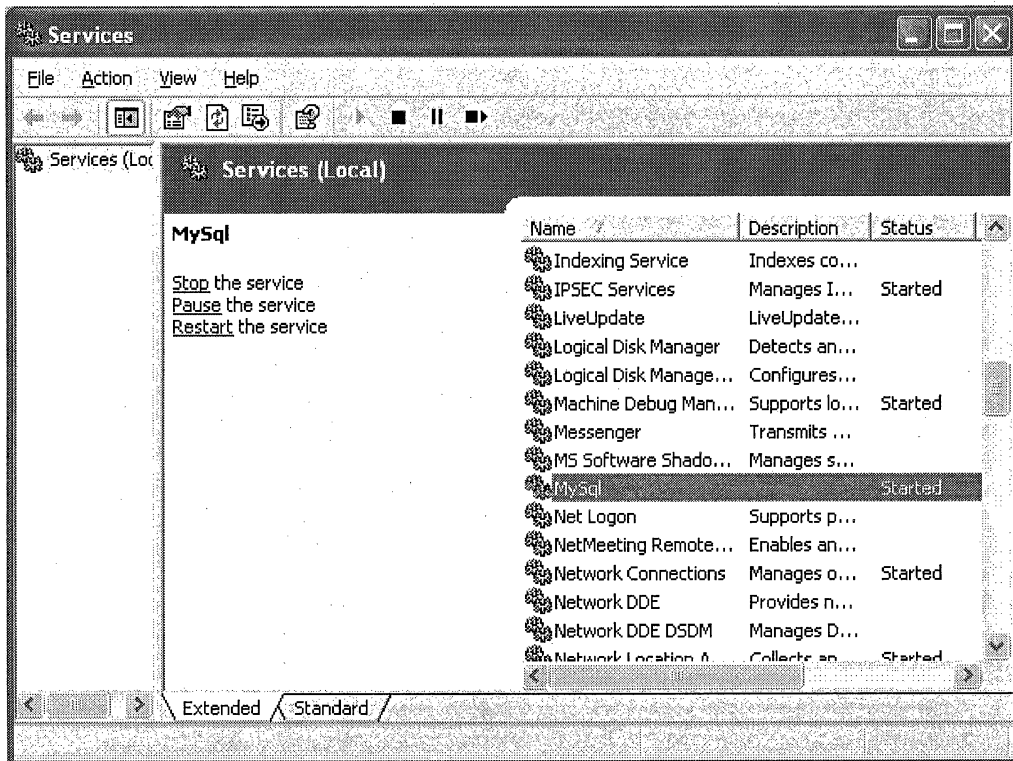


Figure 12. MySQL Windows Service

CHAPTER SIX

CONCLUSIONS

6.1 Summary

The Phi Beta Delta application is developed as a website that is easy to use and administer. The site has links that are easy to navigate. Different levels of access are provided to the users of the website, based on access privileges. A site administrator with the right privileges can access the website and make modifications to the website. The website uses technologies that are open and free. These include the Java programming language, Java 2 Enterprise Edition technologies, Apache Tomcat Servlets engine and MySQL database.

6.2 Enhancements/Improvements

The Phi Beta Delta application developed as part of this project can be enhanced by adding additional functionality that would be useful for end-users and administrators of the web-site. These include support for archiving information from previous years and creating group mailing lists. Other enhancements include providing administrator the ability to upload documents, images and other relevant files that can be picked up by the application automatically.

MySQL is used as the back-end database for the Phi Beta application. MySQL supports high performance, secure ACID (Atomicity, Consistency, Isolation, Durability) compliant transactions. This enables the use of MySQL in high performance mission-critical applications [16]. Other open-source databases such as PostgreSQL are also worth considering when the requirements of the Phi Beta applications change such as support for geographical data including campus maps, directions etc. [17].

6.3 Conclusions

The new website of the Gamma Lambda Chapter of the Phi Beta Delta International Honor Society developed in this project is a great improvement over the existing website in its functionality and ease of use. Furthermore, the site is very easy to administer by a site administrator. The use of open source technologies makes the cost of development and hosting of this site free.

APPENDIX A
SOFTWARE INSTALLATION

SOFTWARE INSTALLATION

This section describes the various aspects involved in installing and maintaining the code and the database of the Phi Beta Delta application on the Windows operating system.

The installation steps include installation of Java Development Kit (JDK), Apache Tomcat Servlets engine and MySQL database.

A1 Installation of Java Development Kit (JDK)

Java Development Kit can be downloaded and installed from the Sun Microsystems web site and is available at the following web site

<http://java.sun.com/j2se/1.5.0/install-windows.html>

The installation package is available as a self-installing executable that installs the JDK software bundle.

A2 Installation of Apache Tomcat

Tomcat installation package is available to be downloaded and installed as a windows installer from the following URL.

<http://tomcat.apache.org/download-55.cgi>

The windows executable 'apache-tomcat-5.5.17.exe' is a wizard-based installer with easy-install instructions. The installer installs Tomcat as a windows service.

A3 Installation of MySQL

MySQL database can be downloaded and installed from the MySQL website at the following URL.

<http://dev.mysql.com/downloads/mysql/5.0.html#win32>

The installer is available in a Microsoft installer format for windows.

A4 Setting up The Application

The Phi Beta Delta web application is packaged as a WAR (Web Archive) file. To deploy the application, the WAR file needs to be placed in a directory called 'webapps' of the Tomcat software before starting the Tomcat service. By default the Tomcat runs at port 8080 and the web application can be accessed at the following URL once it is deployed.

<http://<hostname>:8080/hibeta/>

APPENDIX B
SAMPLE SOURCE CODE

SAMPLE SOURCE CODE

```
public class Member extends BaseObject{
    private String _name="";
    private int _memberId;
    private String _idString;
    private String _firstName;
    private String _lastName;
    private String _middleInitial;
    private String _memberType;
    private java.util.Date _memberSince;
    private String _memberSinceString;
    private String _memberStatus;
    private String _emailAddress;
    private String _address1;
    private String _address2;
    private String _phone1;
    private String _phone2;
    private String _gender;

    public Member(){

    }
    public void setMemberId(int memberId){
        _memberId=memberId;
    }
    public int getMemberId(){
        return _memberId;
    }

    public String getMemberSinceString(){
        return _memberSinceString;
    }

    public void setMemberSinceString(String memberSinceString){
        _memberSinceString=memberSinceString;
    }
    public String getIdString(){
        return Integer.toString(_memberId);
    }
    public String getFirstName() {
        return _firstName;
    }
}
```



```

public void setFirstName(String _firstName) {
    this._firstName = _firstName;
}

public String getLastName() {
    return _lastName;
}

public void setLastName(String _lastName) {
    this._lastName = _lastName;
}

public void setName(String name){
    _name=name;
}
public String getName() {
    _name= _lastName+" " + _firstName;
    return _name;
}

public String getEmailAddress() {
    return _emailAddress;
}

public void setEmailAddress(String emailAddress) {
    _emailAddress = emailAddress;
}

public String getMiddleInitial() {
    return _middleInitial;
}

public void setMiddleInitial(String _middleInitial) {
    this._middleInitial = _middleInitial;
}

public String getMemberType() {
    return _memberType;
}

public void setMemberType(String _memberType) {
    this._memberType = _memberType;
}

```

```

public java.util.Date getMemberSince() {
    return _memberSince;
}

public void setMemberSince(java.util.Date _memberSince) {
    this._memberSince = _memberSince;
    _memberSinceString=_memberSince.toString();
}

public String getMemberStatus() {
    return _memberStatus;
}

public void setMemberStatus(String _memberStatus) {
    this._memberStatus = _memberStatus;
}

public String getAddress1() {
    return _address1;
}

public void setAddress1(String _address1) {
    this._address1 = _address1;
}

public String getAddress2() {
    return _address2;
}

public void setAddress2(String _address2) {
    this._address2 = _address2;
}

public String getPhone1() {
    return _phone1;
}

public void setPhone1(String _phone1) {
    this._phone1 = _phone1;
}

public String getPhone2() {
    return _phone2;
}

```

```

public void setPhone2(String _phone2) {
    this._phone2 = _phone2;
}
public String getGender() {
    return _gender;
}

public void setGender(String gender) {
    this._gender = gender;
}
}

public class ClubMembersServlet extends javax.servlet.http.HttpServlet {
    private edu.dao.BaseDAO _memberDAO = new edu.dao.BaseDAO();
    protected void doGet(javax.servlet.http.HttpServletRequest
        httpRequest, javax.servlet.http.HttpServletResponse
        httpResponse) throws javax.servlet.ServletException,
        java.io.IOException {
        javax.servlet.RequestDispatcher dispatcher = null;
        try {
            String action = httpRequest.getParameter("action");
            String memberId = httpRequest.getParameter("id");

            if (action == null) {
                java.util.List members = _memberDAO.getAllMembers();

                httpRequest.getSession().setAttribute("memberList", members);
                dispatcher =
getServletContext().getRequestDispatcher("/clubMembers.jsp");
                dispatcher.forward(httpRequest, httpResponse);
            } else {
                if (httpRequest.isUserInRole("ADMIN")) {
                    if (action.equalsIgnoreCase("viewmember")) {
                        int id = memberId != null ? Integer.parseInt(memberId) : 0;
                        System.out.println("id is viewMember");
                        edu.model.Member member = _memberDAO.getMember(id);

                        httpRequest.setAttribute("member", member);
                        dispatcher = getServletContext().getRequestDispatcher
                            ("/editMember.jsp");
                        dispatcher.forward(httpRequest, httpResponse);
                    } else if(action.equalsIgnoreCase("edit")){
                        editMember(httpRequest,httpServletResponse);
                    }
                }
            }
        }
    }
}

```

```

        }else if(action.equalsIgnoreCase("newmember")){
            dispatcher = getServletContext().getRequestDispatcher
                ("/newMember.jsp");
            dispatcher.forward(httpServletRequest, httpServletResponse);
        }else if(action.equalsIgnoreCase("addmember")){
            editMember(httpServletRequest,httpServletResponse);
        }
    } else{
        System.out.println("User in not in Admin ROLE");
    }
}
} catch (Exception e) {
    e.printStackTrace();
    dispatcher = getServletContext().getRequestDispatcher("/error.jsp");
    dispatcher.forward(httpServletRequest, httpServletResponse);
}
}
}

```

```

private void editMember(javax.servlet.http.HttpServletRequest
    httpServletRequest, javax.servlet.http.HttpServletResponse
    httpServletResponse) throws Exception{
    java.util.Map paramMap =httpServletRequest.getParameterMap();

```

```

    String id = httpServletRequest.getParameter("id");
    String type = httpServletRequest.getParameter("type");
    String initial = httpServletRequest.getParameter("initial");
    String email = httpServletRequest.getParameter("email");
    String fName= httpServletRequest.getParameter("firstName");
    String lName = httpServletRequest.getParameter("lastName");
    String status = httpServletRequest.getParameter("status");
    String since = httpServletRequest.getParameter("since");

```

```

    edu.model.Member member = new edu.model.Member();

```

```

    if(id !=null){
        member.setMemberId(Integer.parseInt(id));
    }

```

```

member.setMemberType(type);
member.setFirstName(fName);
member.setLastName(lName);
member.setEmail(email);
member.setMemberStatus(status);
member.setMemberSince(since);

if(paramMap.keySet().contains("save")){
    saveMember(member);
} else if(paramMap.keySet().contains("delete")){
    deleteMember(member);
} else if(paramMap.keySet().contains("add")){
    addMember(member);
}

}

javax.servlet.RequestDispatcher dispatcher =
    getServletContext().getRequestDispatcher("/clubMembers.jsp");
dispatcher.forward(httpServletRequest, httpServletResponse);
}

private void saveMember(edu.model.Member member) throws Exception{
    _memberDAO.updateMember(member);
}

private void deleteMember(edu.model.Member member) throws Exception{
    _memberDAO.deleteMember(member);
}

private void addMember(edu.model.Member member) throws Exception{
    _memberDAO.addMember(member);
}

protected void doPost(javax.servlet.http.HttpServletRequest
    httpServletRequest, javax.servlet.http.HttpServletResponse
    httpServletResponse) throws javax.servlet.ServletException,
    java.io.IOException {
    doGet(httpServletRequest, httpServletResponse);
}

```

REFERENCES

- [1] Dietel, Dietel and Nieto, Internet and World Wide Web-How to Program, Prentice Hall Publishers, 2000.
- [2] Craig Larman, Applying UML and Patterns: An introduction to Object-Oriented Analysis and Design, Prentice-Hall, Inc., 1998.
- [3] Rick Catell, Jim Inscore, J2EE Technology in Practice, Addison-Wesley Publishers, 2001.
- [4] Jason Hunter, Java Servlet Programming, O'Reiley Publishers, 1998.
- [5] www.java.sun.com/
- [6] www.java.com/
- [7] <http://httpd.apache.org>
- [8] George Reese, Database Programming with JDBC and JAVA, O'Reily Publishers, 1999, pp-41-55.
- [9] Marty Hall, Core Servlets and Java Server Pages, Prentice-Hall Publishers, 2000.
- [10] Susan Fowler, GUI Design Handbook, McGraw-Hill Publishers, 2000.
- [11] Christopher Schmitt, Designing CSS Web Pages, New Riders Press, 2002.
- [12] <http://en.wikipedia.org/wiki/HTTPS>
- [13] <http://www.javaworld.com/javaworld/jw-12-1999/jw-12-ssj-jspmvc.html>
- [14] <http://tomcat.apache.org>
- [15] <http://wiki.apache.org/tomcat/PoweredBy>
- [16] http://www.mysql.com/news-and-events/press-release/release_2002_11.html
- [17] <http://builder.com.com/5100-6388-1050671.html>