DEVELOPMENT OF AN XML/JSP BASED HEALTH SURVEY BUILDER TOOL

by

QIN YE, M.D.

A Master's Thesis

Presented to the Division of

Medical Informatics and Outcomes Research

in the Oregon Health & Science University

School of Medicine

in partial fulfillment of

the requirements for the degree of

Master of Science

April, 2002

School of Medicine

Oregon Health & Science University

Certificate of Approval

This certifies that the M.S. thesis of
Qin Ye, M.D.
has been approved.

[redacted]

Christopher Dubay, Ph.D.
Academic Advisor
Chair, Thesis Committee
Assistant Professor, Medical Informatics and Outcomes Research Division

[redacted]

William Hersh, M.D.
Thesis Committee
Chair, Medical Informatics and Outcomes Research Division

[redacted]

Somnath Saha, M.D., MPH
Thesis Committee
Assistant Professor, Department of Medicine

TABLE OF CONTENTS

ACKNOWLEDGEMENTS

With sincere thanks to my thesis committee chair:

Christopher Dubay, Ph.D.

for his consistent support and assistance

and committee members (in alphabetical order):

William Hersh, MD

Somnath Saha, MD, MPH,

for their guidance in developing this project

I would also like to acknowledge the assistance of (in alphabetical order):

Karen Eden, Ph.D.
Warren Harrison, Ph.D.
William Smith, Ph.D.

This project could not have been completed without the
continued love, support and encouragement of my wife,

Nan Hou, RN, MSN

ABSTRACT

This master's thesis describes the development of a web application enabling health researchers to design and implement health surveys and polls. It concentrates on delivering a prototype capable of supporting several key elements of the survey process. The application is implemented using Java and XML including survey design, collecting respondent data and presenting survey results.

This thesis project explores two research questions: 1) how can new technologies such as XML and Java be applied to traditional health survey research to improve and encourage web-based data collection and management in health care? 2) Will the Java Server Page (JSP)-based, Relational Database-supported Web Survey Builder tool allow survey researchers to set up and administrate web-based questionnaires without being overly cumbersome or time-consuming?

The main goal was to achieve a flexible and platform-independent survey development environment. With JSP-supported web services these kinds of systems are possible to implement. As an effective development tool, Java provides versatile tools for building easy-to-use applications. Coupled with the World Wide Web and XML technology, the system delivers a new type of flexibility.

I have developed a prototype that enables clinical researchers performing surveys to design questionnaires and obtain results within a few hours of initiation. Compared to traditional survey implementation methods, this method is considerably faster and also

more efficient because parts of the survey process that have been traditionally performed by developers are automated. In order to find other similar computer-aided survey instruments, a literature review was conducted. The review focused primarily on systems for developing web-based surveys.

A summary of the results from the two major pre-implementation phases (the analysis and design phases), are then given. The analysis and design phases present different approaches to the requirements analysis of the system, as well as alternative design models. Available technologies are also discussed, and arguments supporting the final technical platform are presented.

The implementation phase was described with emphasis on tools used and course of action. It concludes with a discussion of the most serious problems encountered during implementation. Finally a preliminary evaluation of the system is presented and guidelines for future improvements are given.

INTRODUCTION

Health Surveys have become a widely used research method in most of the developed nations of the world. As a research technique in health care and other social sciences, survey research has derived considerable reliability from its prevalent acceptance and use in academic institutions. [15]

With the exploding popularity of World Wide Web (WWW), the use of Hypertext Markup Language (HTML) forms or Web-based surveys are becoming the prevailing methods of gathering survey data. These methods have simplified the data collection process by formatting and saving responses directly into a database for future analysis. Since HTML forms can be made programmable, it is also possible to have real time error checking and correction increasing the accuracy of the data collection process. The formatting capability of HTML allows the creation of easy-to-read and attractive forms that may improve response rates. In addition, the programmability of HTML forms makes it possible to randomly order responses and customize options based on information that the respondent supplies earlier in the survey.

On the other hand, the HTML format does have some critical disadvantages, such as static and inflexible structure, browser incompatibility issues, and messy raw data collection. These problems can make WWW-based survey design and its implementation an unpleasant experience. For example, recreating and reinventing the whole survey page is required each time a simple new questionnaire is desired. Other issues arise with

integrated, mixed health survey research, which combines paper-based survey forms and online versions in order to avoid biased sampling due to lack of internet access for some populations. The document transformation from printable paper format to HTML format is quite complicated. Researchers must design the paper survey and then collect the data with totally different methods, which in turn can consume resources and escalate costs.

This thesis project explores two questions:

1. How can new technology such as Extensible Markup Language (XML) and Java be applied to traditional health survey research to improve and encourage web-based data collection and management in health care?
2. Will the Java Server Page (JSP) based, Relational Database supported Web Survey Builder tool allow survey researchers to set up and administer web-based questionnaires without being overly cumbersome or time-consuming?

Background

Survey research is one of the most important areas of measurement in medical research. As stated by Aday (1996), "Health surveys have been and will continue to be widely applied to provide important information for health care professionals, health care policymakers, public health researchers, private providers, insurers and health care consumers concerned with the planning, implementation and evaluation of health-related programs and policies." [15]

The broad area of survey research includes any measurement procedures that involve asking questions of respondents. A "survey" can be anything from a short paper-and-pencil feedback form to an intensive one-on-one in-depth interview.

Surveys can be divided into two broad categories: questionnaires and interviews. Questionnaires are usually paper-and-pencil instruments that the respondent completes. Interviews are usually completed by the interviewers based on what the respondent says.

Surveys can also be classified by their methods of data collection. Mail, telephone interview, and in-person interview survey collection methods are the most common. Extraction of data from samples of medical and other records is also frequently conducted.

- Mail surveys can be relatively low in cost. As with any other survey, problems exist in the use when insufficient attention is given to ensuring adequate levels of participation. Mail surveys can be most effective when directed at particular groups, such as subscribers to a specialized magazine or members of a professional association.
- Telephone interviews are an efficient method of collecting some types of data and are being increasingly used. They lend themselves particularly well to situations where timeliness is a factor and the length of the survey is limited. On the other hand, phone-based surveys impose less of a task load on the user, but increase

cognitive load by requiring the user to keep all the options in memory [11]. Also, response data usually are entered by humans, which is an error-prone process. Furthermore, respondents cannot always review their responses, and are typically subject to time constraints.

- In-person interviews in a respondent's home or office are much more expensive than mail or telephone surveys. They may be necessary, however, especially when complex information is to be collected.[14]

Some surveys combine various methods. For instance, a survey worker may use the telephone to "screen" or locate eligible respondents (e.g., to locate older individuals who are eligible for Medicare) and then make appointments for an in-person interview.

Since the 1970s, there has been a trend to use computer assisted methods of data gathering. [18, 23] Special OCR/OMR forms, telephone voice menu systems, e-mailed text, Web forms, and mailed interactive software packages have been developed and extensively investigated for capturing electronic data. [20] Most recently, an innovative computer-based technology, audio computer-assisted self-interviewing, has been developed using the latest advances in computer technology.[24]

Many recent publications have reported using the Internet to conduct survey research[3, 8,17,21,22]. Unlike traditional CATI and CAPI surveys, Web-based surveys are self-completion surveys. Web-based surveys offer several advantages over traditional survey methods, including easy access, instant distribution, decreased turn-over time, and reduced costs. Because most designers of health surveys do not have the resources to

support traditionally implemented large-scale surveys, the advantages provided by web-based surveys have increased its popularity in recent years. Other benefits over conventional paper or interview methods that web surveys can provide include:

1. The data are collected in a digital format, which can be easily stored and analyzed.[1]
2. Related information such as research objectives and information on organizations can be linked to the survey form page. [7]
3. Content updating and revising can be easily done as needed. Content updating and revising can provide for interactive improvement in survey design.
4. Response validation and rapid, or even real-time reporting, can be accomplished with scripting language.

Issue	Questionnaire				Interview	
	Group	Mail	Drop-Off	Internet	Personal	Phone
Are visual presentations possible?	Yes	Yes	Yes	Yes	Yes	No
Is privacy a feature?	No	Yes	No	Yes	Yes	Yes
Can the survey be modified during administration?	No	No	No	Yes	Yes	Yes
Are open-ended questions easy to implement?	No	No	No	Yes	Yes	Yes
Is reading & writing needed?	N/A	Yes	Yes	Yes	No	No
Can the survey be administered to ensure completeness of response?	Yes	No	No	Yes	Yes	N/A
Can you explain study in person?	Yes	No	Yes	No	Yes	N/A
Is it low cost?	Yes	Yes	No	Yes	No	No
Are staff & facilities needs low?	Yes	Yes	No	Yes	No	No
Does it give access to dispersed samples?	No	Yes	No	Yes	No	Yes
Does respondent have time to formulate answers?	No	Yes	Yes	Yes	No	No
Is there personal contact?	Yes	No	Yes	No	Yes	No

Table 1. Comparison of general advantages and disadvantages with survey methods[14]

Drawbacks of HTML and scripting languages

Currently, most web-based surveys are stored and transmitted in HTML. Based on Standard Generalized Markup Language (SGML, ISO 8879), HTML defines a single, fixed type of document with markups, which let you describe a common class of simple office-style reports. HTML emphasizes marking up information at the presentation level

instead of its meaning and content. This severely limits HTML in several important respects, especially extensibility, structure, and validation:

- *Extensibility.* HTML does not allow users to mark up their information using their own tags or attributes in order to precisely refer to the meaning of information.
- *Structure.* HTML has very little internal structure, which means that users can easily write valid HTML that does not make sense at all when they consider the semantics of the elements.
- *Validation.* HTML does not support the kind of language specification that allows using applications to check data for structural validity on importation.

XML and Java Server page

What is XML?

XML is a subset of the SGML, unlike HTML, which is an application format based on SGML.[16] SGML is a way to express structure and content in different types of electronic documents. Already an international standard for over a decade, SGML was used to publish very large documents, such as airplane maintenance manuals. [16]

The structure in XML is built up by markup tags and character data in groups that are normally called elements. Each element represents a logical component of the XML document.

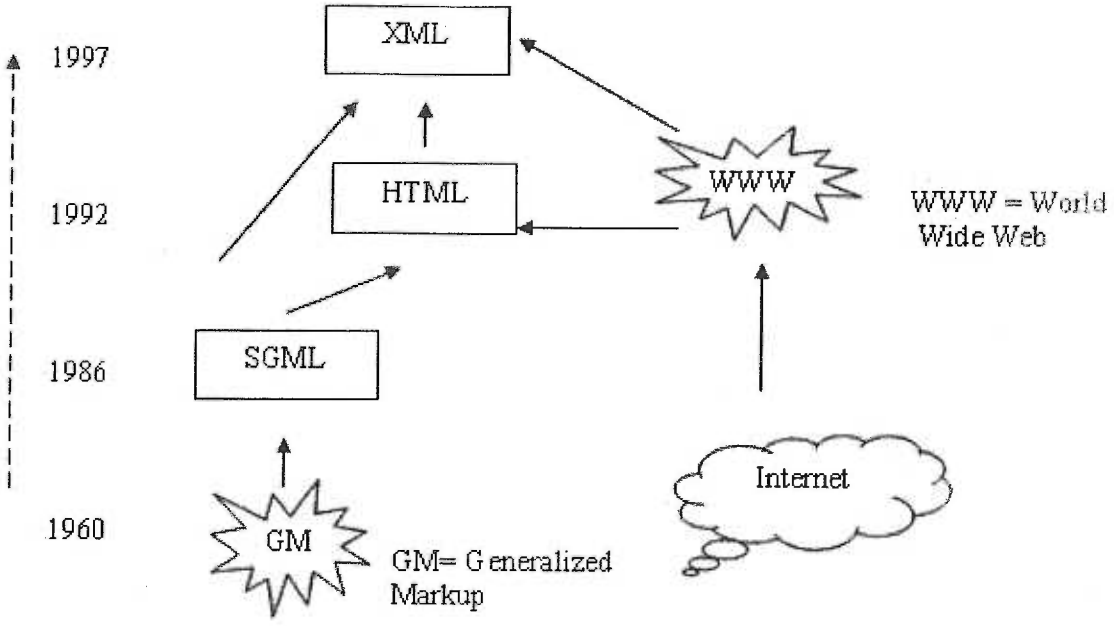


Figure 1: Historical context of Mark up language (Adapted from "The XML companion" by Neil Bradley)

Every XML document consists of elements held together in a logical tree structure. There is always one element that contains all the other elements; this element is called the root element. The tree structure (as illustrated in Figure 2.) of the elements is a vital part of XML and is used when an application wants to read and interpret an XML document.

XML is likely to become the primary vehicle of information interchange on the Web, and it differs from HTML in three major respects: 1) information providers can define new tags and attribute names at will; 2) document structures can be nested to any level of complexity; and 3) any XML document can contain an optional description (DTD) of its grammar for use by applications that need to perform structural validation.

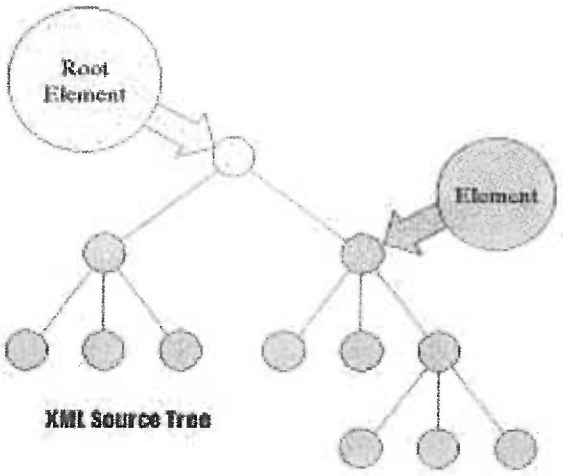


Figure 2: The structure of the XML source tree; the root node and the connected elements.

Why XML?

As stated previously, the limitations of constructing Web pages stem from the fact that HTML uses a fixed number of predefined tags to construct the appearance of a Web page. It is this inability to extend the functionality of the language that has made the development of a new standard for Web publishing desirable.

In recent years, distributed applications in general, and Java in particular, have shown problems caused by the absence of a uniform standard to transfer structured data.[29] Despite the fact that large efforts and huge amounts of money have been put into this area, no publicly approved and used standard has been developed. Problems occur when different applications want to talk to each other, because they almost always use their own incompatible in-house standards for communicating data.[25]

XML was developed to help solve these problems. It offers a structured and consistent way to describe and transfer data. It adds structure and type to information, and allows the information to be stored anywhere on an electronic network, such as the Internet. Thus, data from multiple sources can be aggregated into a single unit of information. Each piece of information has application-specific type and an XML-specific structure. In other words, you'll know how to interpret XML code when you see it.

The goal of data entry should be the complete and accurate electronic representation of raw data for the purpose of communication, processing and archiving. Placing controls upon the practice of data representation, as it concerns a logical class or group of data, is a fundamental tenet of data management. A cooperative effort to establish norms for data representation is necessary if the quality of data as a whole is to improve.

Storing data in XML creates information that can be read by many different types of applications. XML stores data in plain text files, and this simplicity makes it easy to exchange data between incompatible systems. XML supports Unicode, [27] therefore makes the data internationally transportable. Since XML is independent of hardware and software, data are made available to a wider audience of existing (and yet to exist) applications. Other clients and applications can access XML files as data sources, as if they were accessing databases.

XML can also be used to store data in databases.[27] Applications can be written to send and retrieve information from a database, and generic applications can be used to display the data. Finally, XML can be used to create new languages. For instance, the Wireless Markup Language (WML) is used to markup Internet applications for handheld devices like mobile phones. WML is written in XML.

Overview of Java Server Page

Java Server Page (JSP) is an integral part of Java technology in the Java Enterprise 2nd version (J2EE) platform for building applications containing dynamic web content such as HTML, DHTML and XML. [10] JSP technology enables the authoring of web pages which create dynamic content easily with maximum power and flexibility.

In its basic form, JSP is a text-based document that describes how to process a request and create a response (Figure 3 shows the process steps graphically). The application logic is embedded in a simple document as bits of code, which may involve JavaBeans, JDBC objects, etc. The separation of user interface and program logic in a JSP page allows for a very convenient delegation of tasks between web-content authors and developers. It also allows developers to create flexible code that can easily be updated and reused.

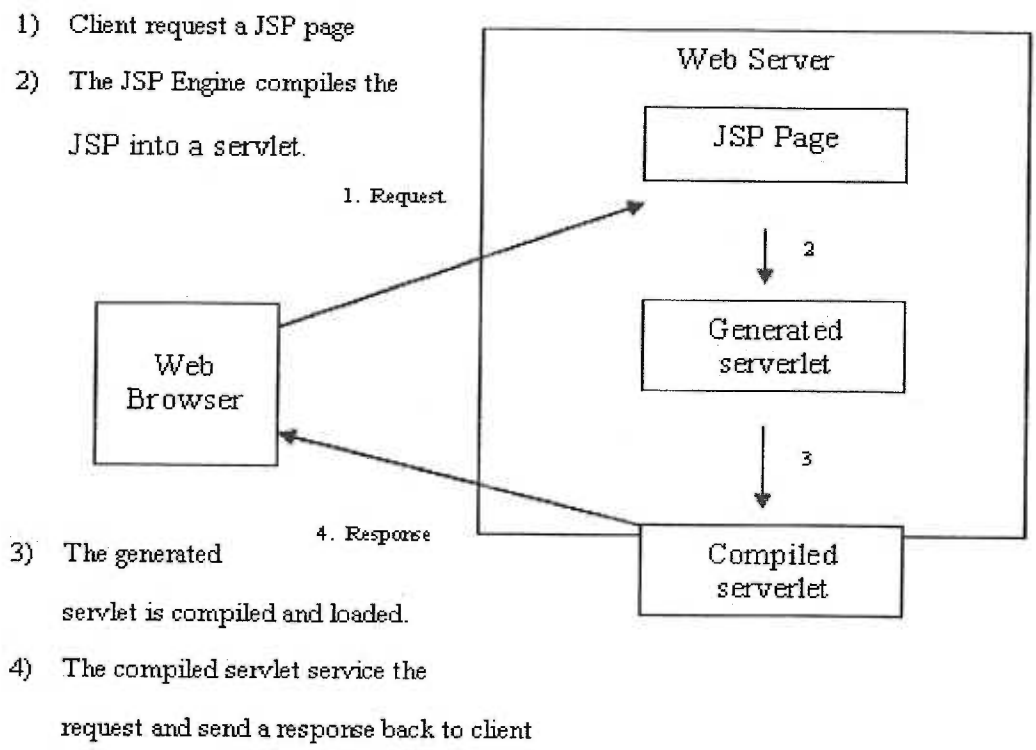


Figure 3. Request- response Process of Java Server Pages

Overall Objectives

The overall objective of this research is to obtain a working knowledge and understanding of how XML and eXtensible Stylesheet Language (XSL) standards can be applied to manage structured or semi-structured survey documents and be dynamically implemented on a web sever or exported to paper based standard forms. The XSL is a separate language for specifying the presentation of XML documents. See reference 27.

The specific aim of this project is to build up an XML-based, JSP driven survey template engine, which is capable of dynamically creating different survey forms,

fetching the input data, and generating output summaries which then can be saved as structured documents or mapped to relational databases.

Anticipated Contribution

The anticipated contribution of this research is in-depth insight into the application of these new technologies in medical research, specifically in the methodology of health survey research. This insight can hopefully help us better plan web-based surveys and expedite the process of planning, designing, and implementing web-based health survey tools.

Although such survey management systems are still in the early research stages, the experience gained through this study will provide practical insight into the requirements for future development.

METHODS

The project consisted of development of a four-tier web application and initial usability testing of that system. The four-tier web application consists of a web-based user interface, an apache web server with servlet container, a set of Java Server Pages to implement index, layout control, and editing functions embedded in several Java Bean components, and mapped XML elements stored in the open-source MySQL Relational Database System. (See Figure 5 for database structure information.)

Requirements Analysis

Over the course of a year, before and during the development of this project, several research meetings have been held with researchers in the Department of Quality Management and the Medical Service at the Portland VAMC to assess the history and limitations of existing patient satisfaction surveys and other health survey research, and the current needs of researchers. These meetings have led to the development of following set of requirements for this thesis project.

As an initial step towards identifying potential solutions, a set of overall functionality and feature requirements was developed.

Functionality requirements

General	
	a. The application should be able to deploy over the Internet with low bandwidth usage b. The software should be easily accessible from multiple departments c. The software components should be reusable d. The software should require little maintenance e. The interface should be GUI-based and easy to understand f. The software should be cross-platform (Mac/Unix/Windows) g. A low-cost solution is desired
Survey design	a. Layout template design editor, ability to use graphics and clips in surveys b. All standard question types (single, multi, grid, open ends) c. Questionnaire libraries and templates d. Re-usable list definitions e. Randomization for unbiased response recordings f. Dynamic text substitution g. Full input validation capabilities
Reporting	a. Real time reporting b. Filtering c. Indexing, averaging, weighting d. Data recoding e. Data export capabilities (MS Excel ,SPSS, ASCII) f. Dynamic report publishing g. Automatic merging of respondent background data and response data

Feature requirements

Question Entity Types

- True /False Questions
- Multiple Choice of Multiple Answers
- Multiple Choice of Single Answers
- Likert Scale Questions
- Semantic Differential Questions
- Open Response Questions
- Numeric Response Questions
- Ranking Order Questions
- Text Block
- Format Block

Administrator Options

- Skip Patterns
- Automatic Ordering
- Description text insertion
- Restrict previous Question Navigation
- View Question Numbers While Taking Survey

Administration Environments

- Conduct surveys on Local Area Networks
- Conduct surveys on Wide Area Networks
- Conduct surveys on The Internet (HTML and JAVA)
- Conduct surveys on standalone or networked Kiosks

System Design

The Survey Builder system was developed on a networked desktop with Red Hat Linux operating system. An Intel Pentium III 1GHz with 40G hard disk served as the Web server, Database server and Development host. See Table 3.

The development environment used for this project was as follows: the operating system was Red Hat Linux 7.2. The IDE used was Sun Forte™ for Java 3.0[10] with J2EE and J2SE support. The RDBMS engine used was MySQL 3.23.47 Red Hat package. The Web Server used for this project was the Apache/Tomcat Web Server running under the system and hardware mentioned above. Tomcat is part of the Apache Software Foundation, developed by the Jakarta Project to provide a reference implementation for the Java servlet and JSP technologies.[30]

Machine	Operating system and other software package
Single networked Intel Pentium III 1Ghz with 40G hard disk and 256MB memory	Red Hat Linux 7.2 or Windows system
	JDK 1.3 or later tested and installed
	Tomcat 4.0 installed and tested
	MySQL 3.23.47
	Type 4 JDBC driver

Table 3. The Hardware and Software configuration for development environment

The project user interface was constructed using Java Server Pages which were driven by JavaBeans running under the Servlet /EJB (Enterprise JavaBeans) container Tomcat. Since the all the components were compliant with Web application archive standard, they are moderately easy to implement and migrate to other servers. Also due to the Java "Write once, Run anywhere" feature, this web application is completely platform independent and extensible.

Description of project development

The initial stage of this project involved an analysis and standardization of health surveys structure. As illustrated as Figure 4, a survey can be compartmentalized into a group of sections, and each section can be further divided into header, introduction and questions, which can in turn become a tree-like hierarchy structure. A tree structure, such as this can be well presented by a set of XML documents.

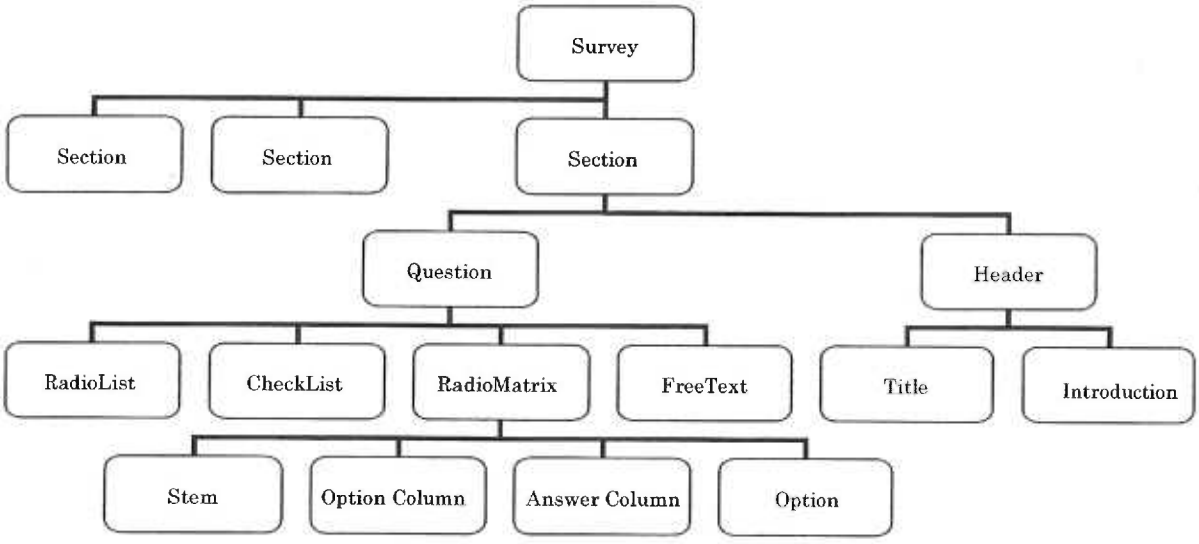


Figure 4: Hierarchy structure of a generic survey

As a web application project, this survey builder consists of several front-end and back-end components. Front-end components provide ease of use and a clear-cut user interface, which includes several java server pages. The back-end system is built upon MySQL, an open-source relational database system (RDBMS), and a set of XML templates. Before the detailed specifications of these modules and their inter-module communications are described, general information about different kinds of database systems will be briefly presented.

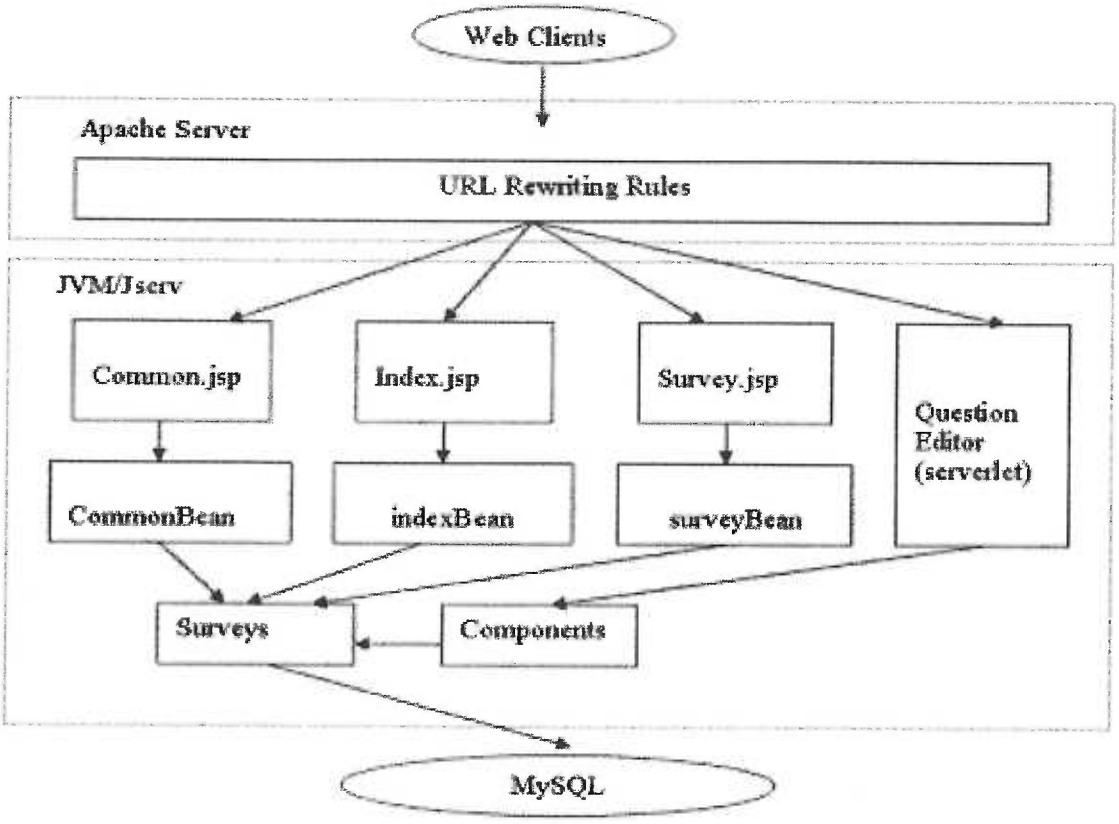


Figure 5: Four-tier structure of proposed application

Database Systems

Even before the arrival of the web, most applications working with a significant amount of data had been using a database system to ease the burden of storage, retrieval and management of data. By delegating this task to a database engine, the web application server can focus on the implementation of critical business logic. There are many classifications of database systems available, such as hierarchical databases, relational databases and object databases, etc. Since the overwhelming majority of today's database applications use relational databases, which is also the choice of this web application, an overview of relational database systems has been presented as following.

Terminology

Term	Explanation
Database	A database is an integrated collection of related data [19]
Database Model	The database model defines all kinds of represented data in terms of their properties, relationships and behavior.
Database Management System	The database management system (DBMS) is a program that manages databases. Thus it realizes the secure, reliable and long-term management of large amounts of data that can be used by many users, and enables access to the data.
Database system	The term database system (DBS) stands for the combination of the DBMS and the database (including the database model).
Application programs	Application programs interact with the DBMS which itself manages and coordinates all access to the databases.
Object-Oriented DBMS	The object-oriented database management-system (OODBMS) is a DBMS with an object-oriented database model [2]

Data Access Language	A data access language (SQL for example) is the language system users employ to communicate with the DBMS[19]

Table 4. DBMS terminology

The Relational Database Model

The relational database model dates back to the year 1970 when it was introduced by E. F. Codd [5]. It does not use pointers to represent relations between data as had been used in earlier database models. In the relational model, data are related to each other exclusively by their content. [4]

Sets of information (called *entities*) that hold data of the same kind are arranged in the form of tables or, in other words, relations. The description of each entity corresponds to one line or row of the respective table, each of those rows being unique within the same relation. As a result, the order of the table's rows does not make any difference.

The columns of the table are called *attributes*. Each attribute has its own name and is always referenced by that name instead of the position. In addition, all attributes have defined ranges of valid values which can be compared with data types using a programming language. It is a very important principle of the relational model that the values of the attributes of a row contain simple values only, or in other words all attribute values are atomic. That means the values cannot be decomposed into smaller pieces.[6]

A *key* of a relation is a subset of the attributes of that relation that meets two conditions:

1. The *unique identification* which guarantees that a table's row can be uniquely referenced by the value of the key.
2. The *absence of redundancies* means that no attribute of the key can be omitted without violating rule 1.

The *relationships* between tables, which are between individual rows or entities of two tables, are generated by storing references to the other table as attribute values in the first table. By way of illustration, say there are two tables *A* and *B*, *A* owning primary key *KeyA*, *B* having primary key *KeyB*. In order to let a certain row of table *B* be related to perhaps some rows of table *A*, the primary key value (*KeyB*) of *B*'s row is stored as attribute values in the concerned rows of table *A*. From table A's point of view, these stored references of *KeyB* are called *foreign key values* because these values actually are key values, thus uniquely identifying *B*'s row, but not key values of table *A*. The term *foreign key* identifies the table's attribute, or subset of attributes, or columns respectively, where the foreign key values are stored.

A relational database model that is designed according to the above definitions may still contain ambiguities and inconsistencies that should be cleared up before implementation. This process is called *normalization.*

The theory of normalization is based on the concept of *normal forms*, starting with the *first normal form*. A relation is in first normal form if every attribute value is an atomic, not further divisible unit of data. As this condition is already included in the

definition of a relation, every relation that is created according to the above definitions is in first normal form.

Every further normal form is based on the previous normal form. In other words, a relation that is in second normal form already is in first normal form or a relation in third normal form is both in first and second normal form and so on. The following normal forms have been defined in this order of increasing precision of their conditions:

- First normal form
- Second normal form
- Third normal form
- Boyce-Codd normal form
- Fourth normal form
- Fifth normal form

A relation is in *second normal form* if it is in first normal form and every non-key attribute is fully functional dependent on the primary key. This means that there must not have a (non-key) attribute of which the value is implied by any other attribute beside the primary key attribute(s).

A relation is in *third normal form* if it is in second normal form and every non-key attribute is not transitively dependent on the primary key. In other words, no non-key attribute is allowed that depends on another non-key attribute which itself depends on the primary key.

Moving from one normal form to the next is always possible by splitting up relations. Of course no information is being lost because the original relation can be restored by joining the new relations.

A relational model in third normal form can still have some inconsistencies that can be eliminated by applying the remaining normal forms. However, such inconsistencies rarely appear and therefore the conversion to the third normal form is sufficient for most applications. In practice a total normalization is not always desirable for the following reason: updating data is simplified in a higher normal form because inconsistencies are prevented, but retrieving data is complicated because more tables must be joined then. This, however, negatively affects the performance of a database application which requires the database developer to carefully consider normalization and performance issues[19].

For this Health Survey Builder system the number of needed attributes is relatively small and the relationships in between are not very complicated, so meeting the conditions of the third normal form will satisfy the Health Survey Builder system's requirements which is why the other normal forms are not treated any further.

Description of the tables

The database tables are the essential components of this web application. For easy access, implementation, and efficiency purposes, the open-source RDBMS MySQL has

been chosen as the database engine. The database interface that provides Java API is the well known, freely available mm JDBC (Java Database Connectivity) driver[12]. The mm JDBC is Type 4 JDBC driver, built upon pure Java, and it communicates with RDBMS directly, which provides maximum compatibility and efficiency. For performance optimization purposes, the database connection pooling mechanism has also been implemented.

The database schema designed for this application is illustrated as Figure 6. Please also refer to Appendix III for detailed information of each table in this relational database.

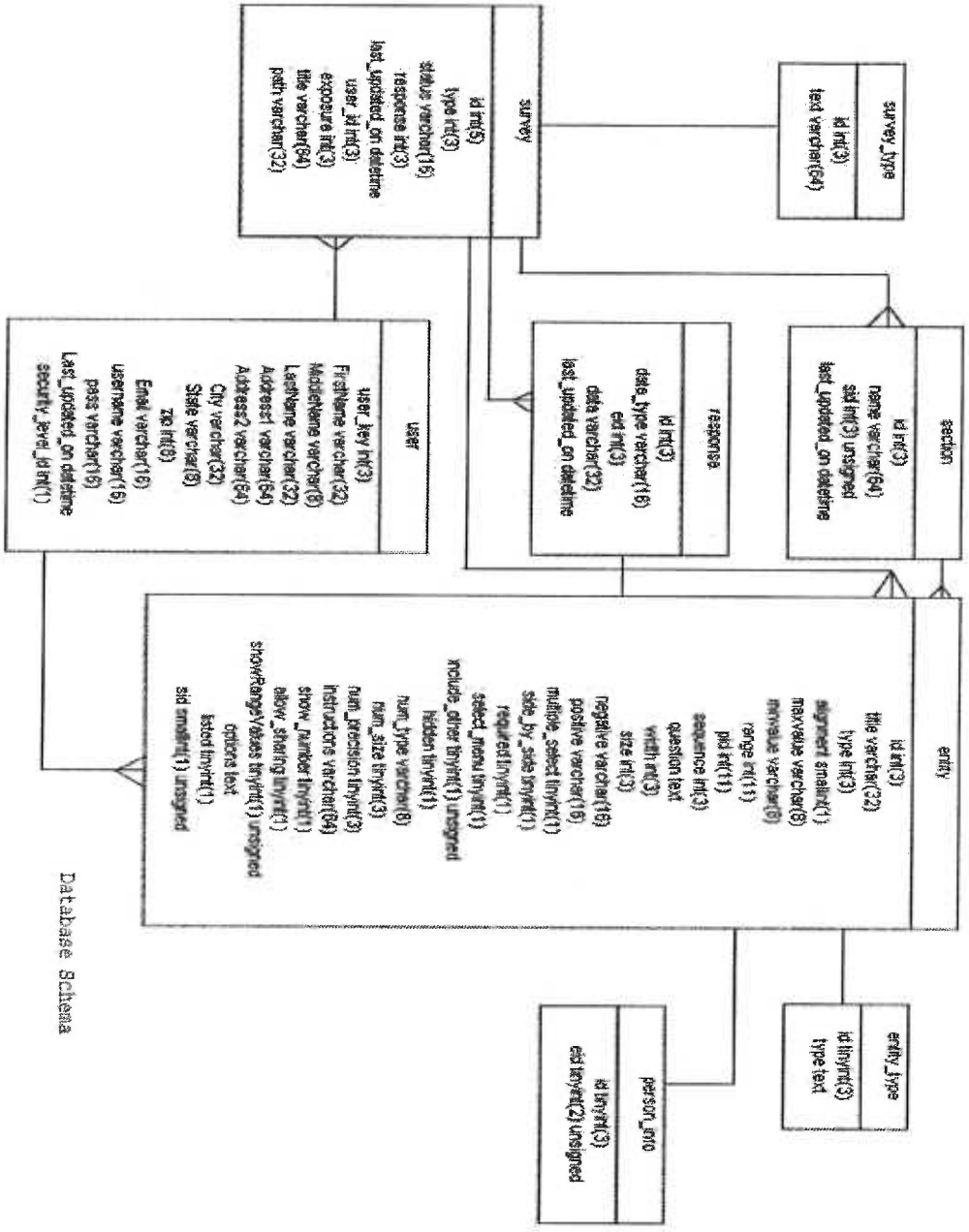


Figure 6. Database Schema of Health Survey Builder Project

Components breakdown

Components	Description
User Interface	To allow easy access and comfortable usage, a web interface will be provided as the front end using Java Server Page technology
Backend process and Template Engine	A set of Java server pages which compiled on the server side will be executed and specific JavaBean component will be called to fulfill user requests and response validation
Questionnaire Designer	Use generic survey components to form a set of XML/DTD/XSL documents, user will be able to create and edit questionnaire in web browser
Index and Metadata	survey templates and documents will be indexed by category, ID, date and keywords as functions provided by index Bean and correlated with JSPs
Results Management	Automatic response collection use mapped XML database to store resulting data Extra modules to provide presentation format for results
Survey report	Real time reports will be provided by a Java Servelet, which can provide descriptive statistics analysis and chart reporting

Table 5. Survey Builder Application Components architecture breakdown

EVALUATION

As stated earlier, the goal of this project was to examine the benefits of web-based survey research with the latest technology, such as XML and JSP, and to test the XML document model. Since the performance evaluation of a web application as large as this can be a project of its own, I decided to carry out a preliminary qualitative assessment, which evaluates the overall usability, user experience and software efficiency.

The primary goal for such an evaluation was to determine if this application can provide an improved development system to help survey researchers design and implement web-based health surveys.

A demonstration of the Health Survey Builder Tool was given to a group of researchers and physicians during a Research-in-Progress Seminar of the Division of General Medicine and Geriatrics at OHSU. In addition to background and technical review of this thesis project, I also gave a step by step demonstration of how to use this tool to design a simple questionnaire. After the demonstration and a feedback session, the participants were asked to fill out a questionnaire. The questionnaire was administered in an electronic format, with questions on the web pages of the application, as well as in a traditional paper format. The survey was piloted with my committee members and several other colleagues within the department for clarity, length, etc.

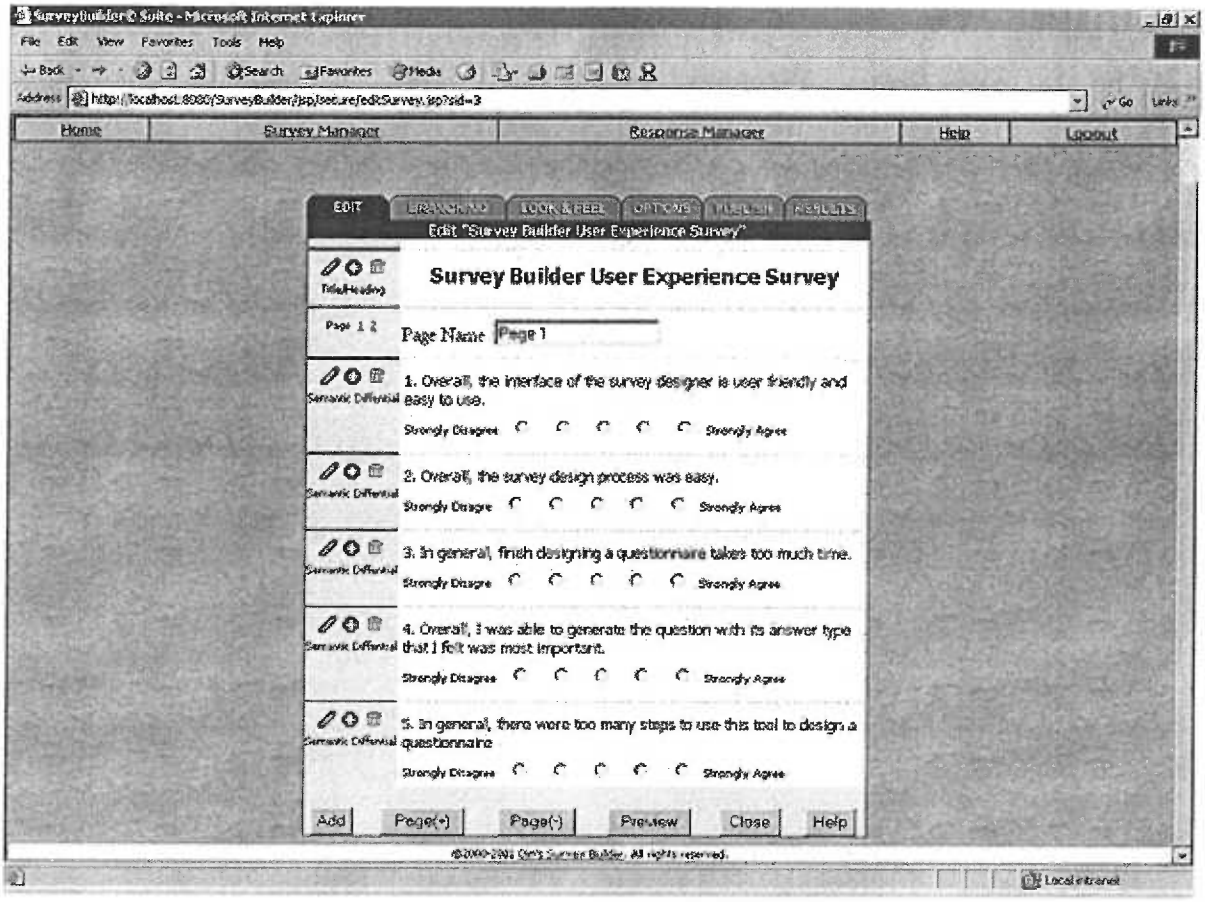


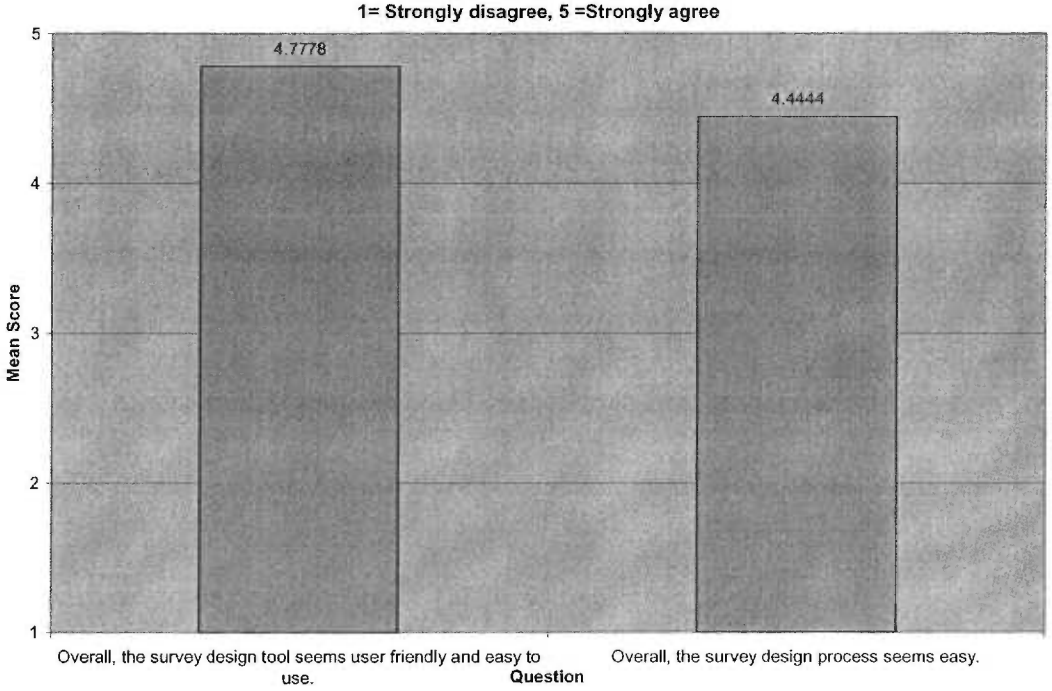
Figure 7. Screen Shot of user experience survey designed by Survey Builder

The goal of this group study was to evaluate the application from a usability perspective, using nine questions to examine aspects such as user interface, complexity, question type representation and general impression.

Figure 8 displays the mean responses to the questions "Overall, the survey design tool seems user friendly and easy to use." and "Overall, the survey design process seems easy."

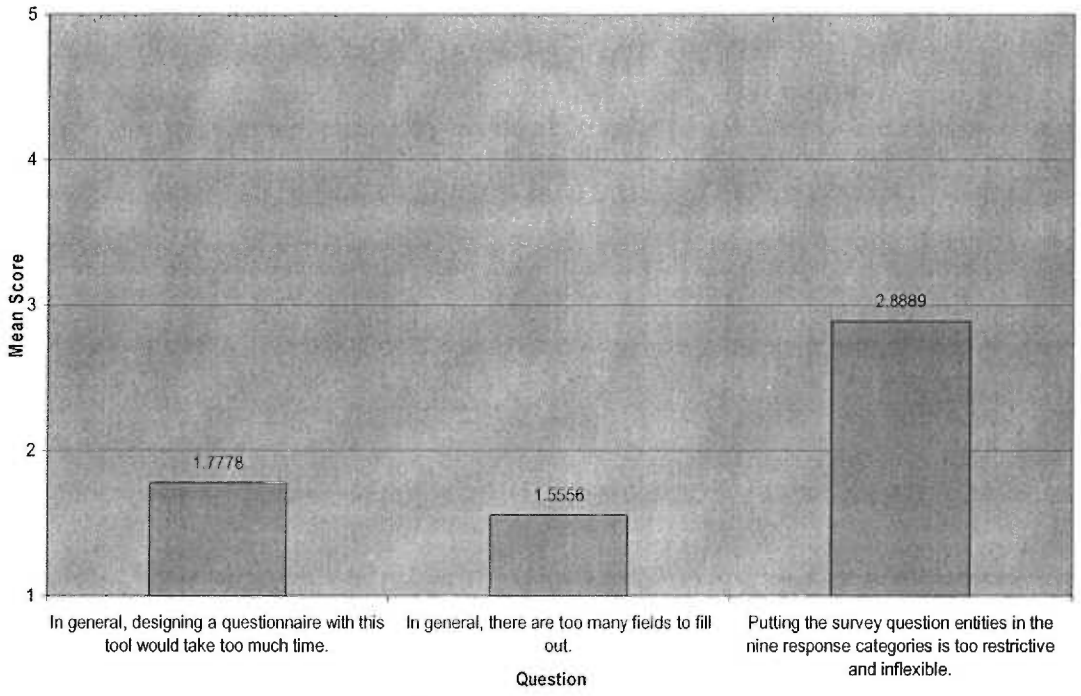
Using a Likert scale with a score of 1 representing *strongly disagree* and 5 representing *strongly agree*. Relatively, users had favorable responses.

Figure 8. Mean Scores
1= Strongly disagree, 5 =Strongly agree



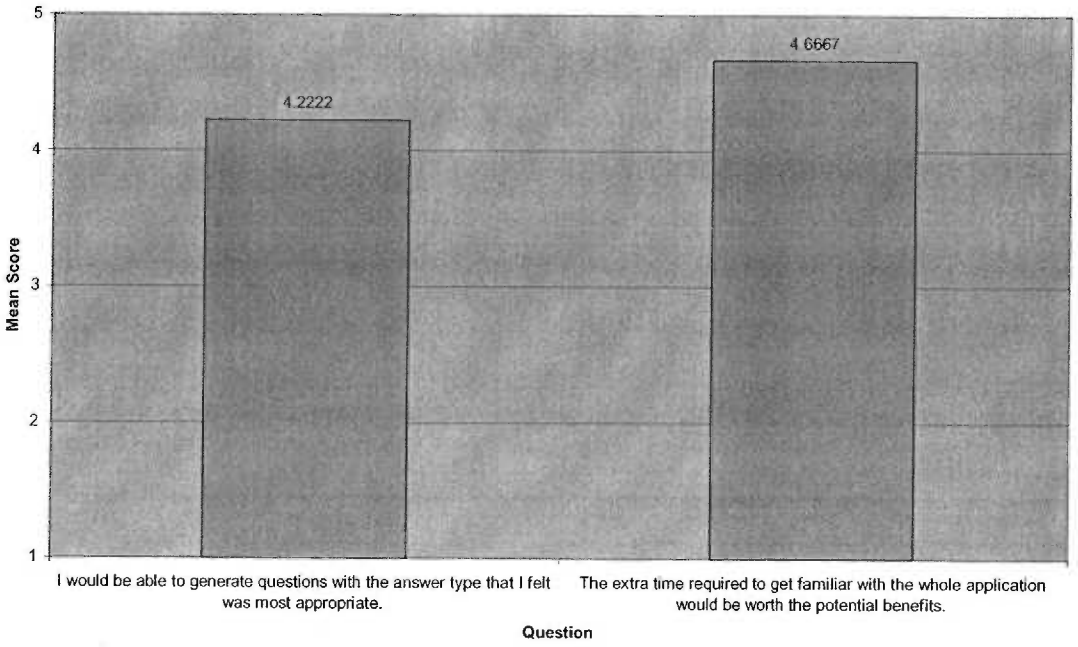
Similarly, participants were asked questions designed to examine how cumbersome the survey design process was, such as "In general, designing a questionnaire with this tool would take too much time," "In general, there were too many fields to fill out," and "Putting the survey question entities in the nine response categories is too restrictive and inflexible." The same Likert scale was used, with responses shown in Figure 9. Users generally did *not* think that too much time was required. The response to the question on flexibility was more equivocal; there was, however, no strong evidence that they felt the nine response categories were too restrictive or inflexible.

Figure 9. Mean Scores
1= Strongly disagree, 5 =Strongly agree



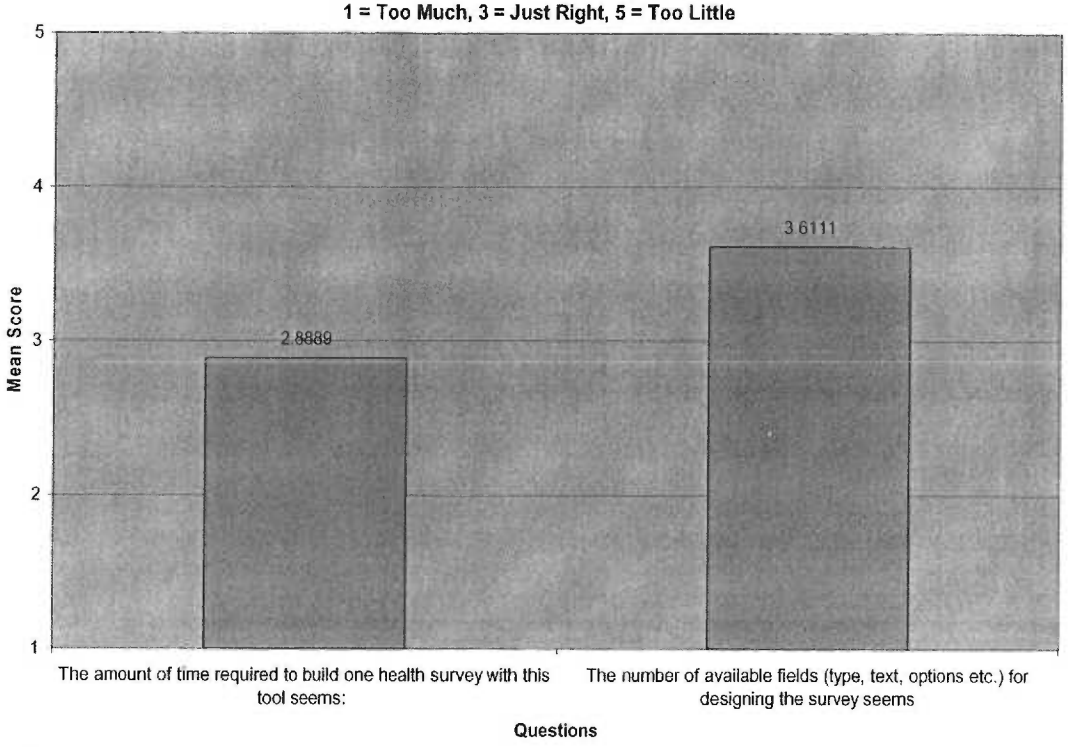
Participants were also given the opportunity to indicate how helpful this tool would be in generating questions with the response type desired and whether the time required getting familiar with the whole application would be worth the potential benefits. The responses to those two questions are shown in Figure 10. The mean responses averaged over the twelve study participants shows that participants also had favorable responses.

Figure 10. Mean Scores
1= Strongly disagree, 5 =Strongly agree



The last two questions in the user experience questionnaire aimed to address the amount of efforts and time need to use this tool compare to participants' previous experiences. The responses to this latter question are shown in Figure 11. As in the previous data, these are mean responses averaged over the eight study participants. Generally, the responses are close to the midline value, labeled "Just Right" on the questionnaire.

Figure 11. Mean Scores
1 = Too Much, 3 = Just Right, 5 = Too Little



DISCUSSION

I have developed a web application that will allow any student, survey researcher and other survey developer to plan, design and administrate web-based questionnaires. It provides them with tools to brainstorm and implement surveys over the Internet without any knowledge of web programming or other technologies. Without such a system, most of these tasks have to be accomplished manually. A manual process is not only inefficient and labor intensive, but also error prone and much less comprehensive compared to this system.

A few commercial tools in use today perform some of these functions, but most are desktop proprietary application and not suitable or integrated for web-based survey implementation.

This Health Survey Builder system has been designed specifically to handle web-based surveys with portable and platform-independence in mind. It has been designed to be able to integrate seamlessly with all the components within the system and provide flexible data format in XML to exchange with external systems.

Future developers can customize this survey builder system and extend its functionality and interfaces for other data collection and analytic purposes.

SUMMARY

Web-based health surveys will become increasingly important in next few years. With an increased percentage of the population using the Internet, the possibilities of sampling the general population will dramatically increase. Since most research in survey methodology still focuses on plain HTML with scripting, my approach in this project, I believe, is novel. This project also demonstrates that a structured and validated survey form in XML format can improve the efficiency of programming and data collection for health survey research.

REFERENCE

1. Afrin LB, Kuppuswamy V, Slater B, Stuart RK. Electronic clinical trial protocol distribution via the WWW: a prototype for reducing costs and errors, improving accrual and saving tress. J Am Med Inform Assoc. 1997;4(1):25 -35.

2. Atkinson M, Bancilhon F, Witt D.J., Dittrich K.R., Maier D., Zdonik S.B. (1989): The Object-Oriented Database System Manifesto (a Political Pamphlet); In: Proc. 1. Intl. Conf. on Deductive and Object-Oriented Databases; Kyoto (Japan).

3. Bloom DE. Technology, experimentation, and the quality of survey data. *Science* 1998;280:847-8.

4. Cattell, Roderic Geoffrey Galton (1991): Object Data Management: Object-Oriented and Extended Relational Database Systems; Reading (MA): Addison-Wesley.

5. Codd, E. F. (1970): A Relational Model of Data for Large Shared Data Banks"; In: Communications of the ACM; Vol. 13; No. 1; pp. 377-387; Baltimore (MD): Association for Computing Machinery.

6. Date C. J., Darwen Hugh (1992): Relational Database Writings 1989-1991; Healdsburg (CA): Addison-Wesley.

7. Dillman D, Tortora RL, Conradt J, Bowker D. Influence of plain vs. fancy design on response rates for Web surveys. In: Proceedings of the Joint Statistical Meetings, Surveys. Methods Section. Alexandria, Va: American Statistical Association, 1998.

8. Eysenbach G, Diepgen TL. Epidemiological data can be gathered with World Wide Web. BMJ.1998; 316(7124):72 .

9. Garratt AM, Ruta DA, Abdalla MI, Buckingham JK, Russell IT. The SF 36 health survey questionnaire: an outcome measure suitable for routine use within the NHS? BMJ.1993; 306:1440 -4.

10. http://java.sun.com/products/jsp/keyfeatures.html

11. http://www.w3.org/pub/WWW/TR/WD-xml-961114.html

12. http://mmmysql.sourceforge.net/

13. Hughes, John G. (1991): Object-oriented databases; Hertfordshire (England): Prentice-Hall International.

14. Louis M. Rea, Richard A. Parker(1992): Designing and Conducting Survey Research: A Comprehensive Guide.

15. Lu Ann Aday (1996): Designing and Conducting Health Surveys : A Comprehensive Guide, Jossey-Bass

16. MacMillan HL. Computer survey technology: a window on sensitive issues. *CMAJ. 161(9):1142, 1999 Nov 2.*

17. Pitkow, James E., and Recker, Margaret M. (1995): "Using the Web as a Survey Tool: Results from the Second WWW User Survey". Journal of Computer and ISDN Systems, 27 (6):809-822.

18. Saris, W.E. (1991): Computer-Assisted Interviewing. Series Quantitative Applications in the Social Sciences, London: Sage University.

19. Sayles, Jonathan (1989): SQL Spoken Here, Wellesley (MA) QED Information Sciences.

20. Schleyer TKL, Forrest JL. Methods for the design and administration of Web-based surveys. J Am Med Inform Assoc.2000; 7:416 -25

21. Sell RL. Research and the Internet: an e-mail survey of sexual orientation. Am J Public Health.1997; 87(2):297 .

22. Suchard MA, Adamson S, Kennedy S. Piloting patient attitudinal surveys on the Web. BMJ.1997; 315:529

23. Tesch R. Computer programs that assist in the analysis of qualitative data: an overview. Qualitative Health Res.1991; 1:309 -25

24. Witt, Karlan J. 1997. Best Practices in Interviewing Via the Internet. Paper Presented at the Sawtooth Software Conference. Seattle, WA.

25. Brett McLaughlin, Mike Loukides (2000): Java and XML, O'Reilly & Associates

26. Charles F. Goldfarb(1991): The SGML Handbook, Clarendon Pr.

27. Neil Bradley(2000): The XML companion, Addison-Wesley Pub Co.

28. Sean McGrath (1998): Rendering XML Documents Using XSL, DDL

29. Simon St. Laurent, Ethan Cerami (1999): Building XML Applications, Computing McGraw-Hill

30. http://jakarta.apache.org/tomcat/index.html

Appendix I: XML document source for generic survey form

Survey.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE survey SYSTEM "/DTDs/survey.dtd">
<survey>
    <section xlink:href="Section.xml"/>
    <section xlink:href="Section2.xml" mandatory="yes"/>
</survey>
```

textArea.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE section SYSTEM "/DTDs/section.dtd">
<section sID="textField">
    <header>
        <title>A single text area</title>
        <intro>
        This is a single text area.
        </intro>
    </header>
    <question>
        <stem>This is now the question.</stem>
        <freeText>
            <stem>This is the extra stem</stem>
            <textInput cols="80" rows="5" type="textArea"/>
        </freeText>
    </question>
</section>
```

CheckList.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE section SYSTEM "/DTDs/section.dtd">
<section sID="checkList">
    <header>
        <title>A single check list</title>
        <intro>
        This is a single check list.
        </intro>
    </header>
    <question>
        <stem>This is now the question.</stem>
        <checkList>
            <stem>This is the extra stem</stem>
            <optionColumn title="Available options"/>
            <option description="Option a"/>
            <option description="Option b"/>
            <otherOption/>
        </checkList>
    </question>
</section>
```

textField.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE section SYSTEM "/DTDs/section.dtd">
<section sID="textField">
    <header>
        <title>A single text field</title>
        <intro>
This is a single text field.
        </intro>
    </header>
    <question>
        <stem>This is now the question.</stem>
        <freeText>
            <stem>This is the extra stem.</stem>
            <textInput cols="80" type="textField"/>
        </freeText>
    </question>
</section>
```

RadioList.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE section SYSTEM "/DTDs/section.dtd">
<section sID="radioList">
    <header>
        <title>A single radio list</title>
        <intro>
        This is a single radio list.
        </intro>
    </header>
    <question>
        <stem>This is now the question.</stem>
        <radioList>
            <optionColumn title="Available options"/>
            <option description="Option a"/>
            <option description="Option b"/>
            <otherOption/>
        </radioList>
    </question>
</section>
```

selectList.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE section SYSTEM "/DTDs/section.dtd">
<section sID="selectList">
    <header>
        <title>A single select list</title>
        <intro>
        This is a single select list.
        </intro>
    </header>
    <question>
        <stem>This is now the question.</stem>
        <selectList>
            <stem>This is the extra stem</stem>
            <option description="Option a"/>
            <option description="Option b"/>
            <otherOption/>
        </selectList>
    </question>
</section>
```

RadioMatrix.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE section SYSTEM "/DTDs/section.dtd">
<section sID="radioMatrix">
    <header>
        <title>A single radio matrix</title>
        <intro>This is a single radio matrix.
        </intro>
    </header>
    <question>
        <stem>This is now the question.</stem>
        <radioMatrix>
            <stem>This is the extra stem.</stem>
            <optionColumn title="Requested options"/>
            <answerColumn title="Answer 1"/>
            <answerColumn title="Answer 2"/>
            <answerColumn title="Answer 3"/>
            <answerColumn title="Answer 4"/>
            <option description="Option 1"/>
            <option description="Option 2"/>
            <option description="Option 3"/>
            <otherOption/>
        </radioMatrix>
    </question>
</section>
```

AlphaMatrix.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE section SYSTEM "/DTDs/section.dtd">
<section sID="alphaMatrix">
    <header>
        <title>A single alpha matrix</title>
        <intro>
        This is a single alpha matrix.
        </intro>
    </header>
    <question>
        <stem>This is now the question.</stem>
        <alphaMatrix allowNA="yes" cols="5" restriction="positive" type="integer">
            <stem>This is the extra stem.</stem>
            <optionColumn title="Requested options"/>
            <answerColumn title="Answer 1"/>
            <answerColumn title="Answer 2"/>
            <answerColumn title="Answer 3"/>
            <answerColumn title="Answer 4"/>
            <option description="Option 1"/>
            <option description="Option 2"/>
            <option description="Option 3"/>
            <otherOption cols="20"/>
        </alphaMatrix>
    </question>
</section>
```

Appendix II: DTD document source for generic survey form

```
<?xml version="1.0"?>
<schema>
    <element name="survey" content="elementOnly">
        <group>
            <element name="section" minOccurs="1" maxOccurs="*"/>
        </group>
    </element>
    <element name="section" content="elementOnly">
        <group order="sequence">
            <element name="header"/>
            <element name="question" minOccurs="1" maxOccurs="*"/>
        </group>
        <attribute name="sID" type="ID" minOccurs="1"/>
    </element>
    <element name="header" content="elementOnly">
        <group order="sequence">
            <element name="title"/>
            <element name="intro"/>
        </group>
    </element>
    <element name="question" content="elementOnly">
        <group order="choice">
            <element name="radioMatrix"/>
            <element name="radioList"/>
            <element name="checkList"/>
            <element name="freeText"/>
            <element name="alphaMatrix"/>
            <element name="selectList"/>
            <element name="combination"/>
        </group>
        <attribute name="qID" type="ID" minOccurs="1"/>
    </element>
    <element name="title" content="textOnly"/>
    <element name="intro" content="textOnly"/>
    <element name="radioMatrix" content="elementOnly">
        <group order="sequence">
            <element name="stem"/>
            <element name="optionColumn"/>
            <element name="answerColumn" minOccurs="1" maxOccurs="*"/>
            <element name="option" minOccurs="1" maxOccurs="*"/>
            <element name="otherOption" minOccurs="0" maxOccurs="1"/>
        </group>
    </element>
    <element name="radioList" content="elementOnly">
        <group order="sequence">
            <element name="stem"/>
            <element name="optionColumn"/>
            <element name="option" minOccurs="1" maxOccurs="*"/>
            <element name="otherOption" minOccurs="0" maxOccurs="1"/>
        </group>
        </element>
```

```
<element name="checkList" content="elementOnly">
        <group order="sequence">
            <element name="stem"/>
            <element name="optionColumn"/>
            <element name="option" minOccurs="1" maxOccurs="*"/>
            <element name="otherOption" minOccurs="0" maxOccurs="1"/>
        </group>
    </element>
    <element name="freeText" content="elementOnly">
        <group order="sequence">
            <element name="stem"/>
            <element name="textInput" minOccurs="1" maxOccurs="*"/>
        </group>
    </element>
    <element name="alphaMatrix" content="elementOnly">
        <group order="sequence">
            <element name="stem"/>
            <element name="optionColumn"/>
            <element name="answerColumn" minOccurs="1" maxOccurs="*"/>
            <element name="option" minOccurs="1" maxOccurs="*"/>
            <element name="otherOption" minOccurs="0" maxOccurs="1"/>
        </group>
        <attribute name="type" type="" default="alpha">
            <datatype source="NMTOKEN">
                <enumeration value="alpha"/>
                <enumeration value="float"/>
                <enumeration value="integer"/>
            </datatype>
        </attribute>
        <attribute name="restriction" type="" default="NA">
            <datatype source="NMTOKEN">
                <enumeration value="positive"/>
                <enumeration value="percent"/>
                <enumeration value="NA"/>
            </datatype>
        </attribute>
    </element>
    <element name="selectList" content="elementOnly">
        <group order="sequence">
            <element name="stem"/>
            <element name="option" minOccurs="1" maxOccurs="*"/>
            <element name="otherOption" minOccurs="0" maxOccurs="1"/>
        </group>
    </element>
    <element name="textInput" content="empty">
        <attribute name="type" type="" default="textField">
            <datatype source="NMTOKEN">
                <enumeration value="textField"/>
                <enumeration value="textArea"/>
            </datatype>
        </attribute>
        <attribute name="title" type="string" minOccurs="1"/>
    </element>
</schema>
```

Appendix III: Database table detailed description

(1) Table 'user'

```
CREATE TABLE `user` (
  `user_key` int (3) NOT NULL auto_increment,
  `FirstName` varchar (32) NOT NULL default '',
  `MiddleName` varchar (8) default NULL,
  `LastName` varchar (32) NOT NULL default '',
  `Address1` varchar (64) NOT NULL default '',
  `Address2` varchar (64) default NULL,
  `City` varchar (32) NOT NULL default '',
  `State` varchar (8) NOT NULL default '',
  `Zip` int (8) NOT NULL default '0',
  `Email` varchar (16) NOT NULL default '',
  `Username` varchar (16) NOT NULL default '',
  `Pass` varchar (16) default NULL,
  `Last_updated_on` datetime NOT NULL default '0000-00-00 00:00:00',
  `security_level_id` int (1) NOT NULL default '1',
  PRIMARY KEY (`user_key`),
  UNIQUE KEY `user_key` (`user_key`)
            )
```

Table Description

user_key ---- the unique user identifier

Username --- the auto generated username

Pass ---- user's password

FirstName, LastName, MiddleName ----- user's name

Address1, Address2, City, Zip, Email ---- user demographic information

Security_level_id ---- unique identifier defines user access privilege

(2) Table `survey`

```
CREATE TABLE `survey` (
 `id` int (5) NOT NULL auto_increment,
 `type` int (3) NOT NULL default '0',
 `status` varchar (16) NOT NULL default 'Active',
 `response` int (3) NOT NULL default '0',
 `last_updated_on` datetime NOT NULL default '0000-00-00 00:00:00',
 `user_id` int (3) NOT NULL default '0',
 `exposure` int (3) default '0',
 `title` varchar (64) NOT NULL default '',
 `path` varchar (32) default NULL,
  PRIMARY KEY (`id`),
  UNIQUE KEY `survey_id` (`id`)
)
```

Table Description

id --- the unique survey identifier

type --- the type of health survey

status --- the edit and design status of survey

response --- the count of survey total responses

user_id --- the unique user identifier (foreign key)

exposure --- the accessible status of survey

title --- the title of the survey

path ---- the physical path of generated survey

The created health survey general information including type, status, response rate and title from each user are stored at this table.

(3) Table `survey_type`

```
CREATE TABLE `survey_type` (
 `id` int(3) NOT NULL default '0',
 `text` varchar(64) NOT NULL default '',
 PRIMARY KEY  (`id`)
       )
```

Table Description

id ----- unique survey type identifier (Primary key)

text ---- survey type

The health survey type can be subcategorized, extended and stored into this table, the user interface will automatically present them to the user. It can also be customerized based on user's specific needs.

(4) Table `section`

```
CREATE TABLE `section` (
 `id` int (3) NOT NULL auto_increment,
 `name` varchar (64) NOT NULL default '',
 `sid` int(3) unsigned NOT NULL default '0',
 `last_updated_on` datetime default '0000-00-00 00:00:00',
 PRIMARY KEY (`id`),
 UNIQUE KEY `id` (`id`)
)
```

Table Description

id ----- unique section identifier

name ---- section name

sid ----- unique survey identifier (foreign key)

last_updated_on --- timestamp of last update

The section table basically defines the pages and their question elements of each survey. By using a separate table, the specific entity or question can be retrieved by

(5) Table `entity`

```
CREATE TABLE `entity` (
 `id` int(3) NOT NULL auto_increment,
 `title` varchar(32) default NULL,
 `type` int(3) default NULL,
 `alignment` smallint(1) default NULL,
 `maxvalue` varchar(8) default NULL,
 `minvalue` varchar(8) default NULL,
 `range` int(11) default NULL,
 `pid` int(11) default NULL,
 `sequence` int(3) default '1',
 `question` text,
 `width` int(3) default NULL,
 `size` int(3) default NULL,
 `negative` varchar(16) default NULL,
 `positive` varchar(16) default NULL,
 `multiple_select` tinyint(1) default NULL,
 `side_by_side` tinyint(1) default NULL,
 `required` tinyint(1) default NULL,
 `select_menu` tinyint(1) default NULL,
 `include_other` tinyint(1) unsigned default NULL,
 `hidden` tinyint(1) default NULL,
 `num_type` varchar(8) default NULL,
 `num_size` tinyint(3) default NULL,
 `num_precision` tinyint(3) default NULL,
 `instructions` varchar(64) default NULL,
 `show_number` tinyint(1) default '0',
 `allow_sharing` tinyint(1) default '0',
 `showRangeValues` tinyint(1) unsigned default '0',
 `options` text,
```

```
  `listed` tinyint(1) default NULL,
  `sid` smallint(1) unsigned default NULL,
  PRIMARY KEY  (`id`)
        )
```

The entity table is the core part of this database schema. It defines all the characteristics of each type of entity. It can be further normalized, but due to the scope of the project, keep all the related information of each entity in one table is the most effective choice.

(6) Table `entity_type`

```
CREATE TABLE `entity_type` (
  `id` tinyint(3) NOT NULL auto_increment,
  `type` text NOT NULL,
  PRIMARY KEY (`id`),
  UNIQUE KEY `id` (`id`)
)
```

Table description

Id ---- the unique entity type identifier

Type ---- the context of entity type

The entity type table illustrates all the question type discussed in feature requirement. It pre-defines nine question types and three format types.

Appendix IV Selected source code of Java Serve Pages

Login.jsp

```
<%@ include file="Common.jsp" %>
<%!
//
//  Filename: Login.jsp


static final String sFileName = "login.jsp";

%>

<%

boolean bDebug = false;

String sAction = getParam( request, "FormAction");
String sForm = getParam( request, "FormName");
String sLoginErr = "";

java.sql.Connection conn = null;
java.sql.Statement stat = null;
String sErr = loadDriver();
conn = cn();
stat = conn.createStatement();
if ( ! sErr.equals("") ) {
 try {
  out.println(sErr);
 }
 catch (Exception e) {}
}
if ( sForm.equals("Login") ) {
  sLoginErr = LoginAction(request, response, session, out, sAction, sForm, conn, stat);
  if ( "sendRedirect".equals(sLoginErr)) return;
}

%>
<html>
<head>
<title>Login</title>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
</head>
<body  bgcolor=#FFFFFF topmargin=0 leftmargin=0 marginheight=0 marginwidth=0>
<table border=0 cellspacing=0 cellpadding=0 width=100% height=80% background="images/bg.gif">
  <tr>
   <td valign=top height="542">
<jsp:include page="banners/top_banner.html" flush="true"/>
 <table border=0 cellpadding=3 width=100% height="525">
<tr>
      <td VALIGN="top" align="middle" width="246">
            <p> </p>
```

```
<% Login_Show(request, response, session, out, sLoginErr, sForm, sAction, conn, stat); %>
<BR>
        <BR><BR>
       </td>
       <td width="54"> </td>
       <td valign=top width="700" >
        <p align="right"><%=java.text.DateFormat.getDateInstance().format(new java.util.Date()
)%></p>
        <p class="blueHeader">Not a registered user?</p>
        <p><br>
         <a href=register/index.jsp><img src="images/registerHere_icon.gif" BORDER="0"></a>
         <BR>
        </p>
       </td>
       <td width="3"> </td>
         </tr>
</table>
</td></tr>
</table>

<jsp:include page="banners/footer.html" flush="true"/>
<center>
 <font face="Arial"><small></small></font>
</center>
</body>
</html>
<%
if ( stat != null ) stat.close();
if ( conn != null ) conn.close();
%>
<%!

 String LoginAction(javax.servlet.http.HttpServletRequest request,
javax.servlet.http.HttpServletResponse response, javax.servlet.http.HttpSession session,
javax.servlet.jsp.JspWriter out, String sAction, String sForm, java.sql.Connection conn,
java.sql.Statement stat) throws java.io.IOException {
   String sLoginErr = "";
   try {
    final int iloginAction = 1;
    final int ilogoutAction = 2;
    String transitParams = "";
    String sQueryString = "";
    String sPage = "";
    String sSQL="";
    int iAction = 0;

    if ( sAction.equals("login") )  iAction = iloginAction;
    if ( sAction.equals("logout") ) iAction = ilogoutAction;
```

```
switch (iAction) {
      case iloginAction: {
       // Login action

       String sLogin = getParam( request, "Login");
       String sPassword = getParam( request, "Password");
       java.sql.ResultSet rs = null;
       rs = openrs( stat, "select user_key, security_level_id , FirstName, LastName from user where
username =" + toSQL(sLogin, adText) + " and pass=" + toSQL(sPassword, adText));

       if ( rs.next() ) {
        // Login and password passed
            String uid = rs.getString(1);
        session.setAttribute("UserID", uid);


        session.setAttribute("UserRights", rs.getString(2));
                              String fullname = rs.getString(3)+" " + rs.getString(4);

                              session.setAttribute("FullName", fullname);
        sQueryString = getParam( request, "querystring");
        sPage = getParam( request, "ret_page");
        if ( ! sPage.equals(request.getRequestURI() ) && ! "".equals(sPage)) {
         try {
          if ( stat != null ) stat.close();
          if ( conn != null ) conn.close();
         }
         catch ( java.sql.SQLException ignore ) {}
         response.sendRedirect(sPage + "?" + sQueryString);
         return "sendRedirect";
        }

        else {
         try {
          if ( stat != null ) stat.close();
          if ( conn != null ) conn.close();
         }
         catch ( java.sql.SQLException ignore ) {}
         response.sendRedirect("/SurveyBuilder/jsp/secure/index.jsp?uid="+uid);
         return "sendRedirect";
        }
       }
       else sLoginErr = "Login or Password is incorrect.";
       rs.close();

       break;
      }
```

```
case ilogoutAction: {
        // Logout action

        session.setAttribute("UserID", "");
        session.setAttribute("UserRights", "");

        break;
       }
      }
    }
    catch (Exception e) { out.println(e.toString()); }
    return (sLoginErr);
  }

void Login_Show(javax.servlet.http.HttpServletRequest request,
javax.servlet.http.HttpServletResponse response, javax.servlet.http.HttpSession session,
javax.servlet.jsp.JspWriter out, String sLoginErr, String sForm, String sAction, java.sql.Connection
conn, java.sql.Statement stat) throws java.io.IOException {
    try {


      String sSQL="";
      String transitParams = "";
      String sQueryString = getParam( request, "querystring");
      String sPage = getParam( request, "ret_page");

      out.println("    <table border=\"1\" cellspacing=\"0\" >");
      out.println("     <tr>\n     <td><img src=\"images/logIn_header.gif\" BORDER=\"0\"></td>\n
</tr>");

      if ( sLoginErr.compareTo("") != 0 ) {
       out.println("<tr>\n <td>"+sLoginErr+"</td>\n     </tr>");
      }
      sLoginErr="";
            out.println("<tr><td VALIGN=\"top\">");
      out.println(" <form action=\""+sFileName+"\" method=\"POST\">");
      out.println(" <input type=\"hidden\" name=\"FormName\" value=\"Login\">");
      if ( session.getAttribute("UserID") == null || ((String)
session.getAttribute("UserID")).compareTo("") == 0 ) {
       // User did not login
       out.println("   Username:<br><input type=\"text\" name=\"Login\" maxlength=\"50\"
value=\""+toHTML(getParam( request, "Login"))+"\"><br>");
       out.println("   Password<br><input type=\"password\" name=\"Password\"
maxlength=\"50\"><br>");
       out.print("  <br><input type=\"hidden\" name=\"FormAction\" value=\"login\"><input
type=\"submit\" value=\"Login\"> <br>");
       out.println("<input type=\"hidden\" name=\"ret_page\" value=\""+sPage+"\"><input
type=\"hidden\" name=\"querystring\" value=\""+sQueryString+"\">     </form>\n ");
      }
```

```
else {
      // User logged in
      String sUserID = dLookUp( stat, "users", "user_login", "user_id =" +
session.getAttribute("UserID"));
      out.print(" <font>"+sUserID+"  "+"</font><input type=\"hidden\"
name=\"FormAction\" value=\"logout\"/><input type=\"submit\" value=\"Logout\"/>");
      out.print("<input type=\"hidden\" name=\"ret_page\" value=\""+sPage+"\"><input
type=\"hidden\" name=\"querystring\" value=\""+sQueryString+"\">");
      out.println("   </form>\n     ");
     }
     out.println(" </td></tr></table>" );


   }
   catch (Exception e) { out.println(e.toString()); }
  }
%>
```

Process_section.jsp

```
<%@ include file="Common.jsp" %>

<%
//=================================
// Obtain DB connection and statement
//---------------------------------
java.sql.Connection conn = null;
java.sql.Statement  stat = null;
String sErr = loadDriver();
if ( ! "".equals(sErr) ) {
 try {
   out.println(sErr);
 }
 catch (Exception e) {}
}
conn = cn();
stat = conn.createStatement();

//Save the name of the form and type of action into the variables
//---------------------------------
String sAction = getParam(request, "FormAction");
String sForm ="editForm";

//=================================
%>

<%
java.sql.ResultSet rs = null;
String sid = (String) session.getAttribute("sid");
String pid = "";
String sql = "";
if (request.getParameter("addPage") != null) {
response.sendRedirect("addSection.jsp");
}
else if (request.getParameter("deletePage") != null)
{
}
else if (request.getParameter("preview")!= null) {
response.sendRedirect("preview_survey.jsp?sid=" + sid);
}
else if (request.getParameter("add")!= null) {

String page_name = request.getParameter("page_name");

try {
sql = "insert into section ( "+
            "name," +
            "sid)" +
            "values (" +
            toSQL(page_name, Text_TYPE) + "," +
            toSQL(sid, Number_TYPE) +
            ")";
```

```
%>
stat.executeUpdate(sql);
} catch (Exception e) {out.println(e.toString()); }
try{
sql = "select id from section where sid =" + sid+ " and name=" + toSQL(page_name, Text_TYPE);
rs = stat.executeQuery(sql);
   } catch (Exception e) {out.println(e.toString()); }

if(rs != null){
if (rs.next() != false) {
rs.next();
pid = rs.getString("id");
                                    }
            }

 response.sendRedirect("editSurvey.jsp?sid="+ sid + "&pid=" + pid);
}

%>
```

editEntity.jsp

```
<%@ include file="Common.jsp" %>

<%!

//=================================
// Save Page and File Name available into variables
//---------------------------------
static final String survey_manager = "survey_manager.jsp";
static final String edit_survey = "survey_manager.jsp";
static final String add_survey = "AddSurvey.jsp";
//=================================

%>
<%
//=================================
// Obtain DB connection and statement
//---------------------------------
java.sql.Connection conn = null;
java.sql.Statement  stat = null;
conn = cn();
stat = conn.createStatement();
%>
<html>
<head>
<title>SurveyBuilder&copy; Suite Edit </title>
<link rel="stylesheet" type="text/css" href="../styles/stylesheet.css">
</head>
<body  bgcolor="#FFFFFF" topmargin=0 leftmargin=0 marginheight=0 marginwidth=0>
<table border=0 cellspacing=0 cellpadding=0 width="100%" height="80%"
background="../images/bg.gif">
<tr>
<td valign=top height="542">
<%@ include file="../banners/secure_banner.html" %>
<table border=0 cellpadding=3 width="100%" height="525">
<tr>
<td>
<p><center>
<%

String sid = (String) session.getAttribute("sid");
String pid = request.getParameter("pid");
String eid = request.getParameter("eid");
String e_type = request.getParameter("entity_type");
java.sql.ResultSet rs = null;

String question = "Write your question here...";
String options = null;
String size = null;
String width = null;
String range = null;
String negative = null;
String positive = null;
```

```
String sql = "select range, question, width, size, negative, positive, options, title from entity where id =
" + eid;

rs = openrs( stat, sql);

 if (rs !=null )
  {
        rs.next();
     range = rs.getString("range");
     question = rs.getString("question");
     width = rs.getString("width");
     size = rs.getString("size");
     negative = rs.getString("negative");
     positive = rs.getString("positive");
     options = rs.getString("options");
     header = rs.getString("title");
  }
}


int i = stringToInt(e_type, 0);

switch(i) {

case 1:
%>
<%@ include file="templates/true_false.jsp" %>
<%

break;

case 2:
%>
<%@ include file="templates/multi_choice.jsp" %>
<%

break;

case 3:
%>
<%@ include file="templates/likert.jsp" %>
<%
break;

case 4:
%>
<%@ include file="templates/differential.jsp" %>
<%
break;

case 5:
%>
<%@ include file="templates/open_response.jsp" %>
<%
break;
```

```
case 6:
%>
<%@ include file="templates/numeric_response.jsp" %>
<%
break;

case 7:
%>
<%@ include file="templates/rank_order.jsp" %>
<%
break;

case 8:
%>
<%@ include file="templates/personal_info.jsp" %>
<%
break;

case 9:
%>
<%@ include file="templates/text_block.jsp" %>
<%
break;

case 10:
%>
<%@ include file="templates/header.jsp" %>
<%
break;

case 11:
%>
<%@ include file="templates/hr.jsp" %>
<%
break;

case 12:
response.sendRedirect("process_entity.jsp?entity_type=12&pid="+ pid+"&listed=1" );
break;
}
%>

</center>
</td>
<td width="3"> </td>
</tr>
</table>
</td></tr>
</table>
<jsp:include page="../banners/footer.html" flush="true"/>
</body></html>
```

Appendix V Selected Source Code of Application wide Java Beans

SurveySessionBean.java interface

```
import java.util.*;
import java.sql.*;
import javax.ejb.*;

public class SurveySessionBean implements SessionBean {

 SessionContext sessionContext;
 String driver, dbURL, user, passwd, tableName;
 // SessionBean methods
 public void setSessionContext (SessionContext sc) {
  // called first by container after Class.newInstance ()
  this.sessionContext = sc;
 }

 public void ejbRemove () {
  // called before container removes bean instance
 }

 public void ejbActivate () {
  // called after bean is activated to allow resource setup
 }

 public void ejbPassivate () {
  // called before bean is passivated to allow resource cleanup
 }

 // SurveySessionServerHome method
 public void ejbCreate (String d, String db, String u, String p, String t) {
  // called second by container
  this.driver = d;
  this.dbURL = u;
  this.passwd = p;
  this.tableName = t;
 }

 // SurveySession business methods
 Hashtable loadAllThreads () {
  // open db connection
  // return a Hashtable with the threads and empty Vectors
 }

 Vector loadThreadSurveys (String t) {
  // open db connection
  // return a Vector with all the articles from thread t
 }
 boolean addSurvey (String art, String t) {
  // open db connection
  // post to the database
  return true;
 }          }
```

FormBean.java

```
import java.util.*;
public class FormBean {
        private String firstName;
        private String lastName;
        private String email;
        private String userName;
        private String password1;
        private String password2;
        private String zip;
        private String notice;
        private Hashtable errors;

        public boolean validate() {
                boolean allOk=true;
                if (firstName.equals("")) {
                        errors.put("firstName","Please enter your first name");
                        firstName="";
                        allOk=false;
                }
                if (lastName.equals("")) {
                        errors.put("lastName","Please enter your last name");
                        lastName="";
                        allOk=false;
                }
                if (email.equals("") || (email.indexOf('@') == -1)) {
                        errors.put("email","Please enter a valid email address");
                        email="";
                        allOk=false;
                }
                if (userName.equals("")) {
                        errors.put("userName","Please enter a username");
                        userName="";
                        allOk=false;
                }
                if (password1.equals("") ) {
                        errors.put("password1","Please enter a valid password");
                        password1="";
                        allOk=false;
                }
                if (!password1.equals("") && (password2.equals("") ||
!password1.equals(password2))) {
                        errors.put("password2","Please confirm your password");
                        password2="";
                        allOk=false;
                }
    if (zip.equals("") || zip.length() !=5 ) {
                        errors.put("zip","Please enter a valid zip code");
                        zip="";
                        allOk=false;
                } else {
```

```
                try {
                        int x = Integer.parseInt(zip);
                    }
                    catch (NumberFormatException e) {
                            errors.put("zip","Please enter a valid zip code");
                            zip="";
                            allOk=false;
                }
}
            return allOk;
 }
 public String getErrorMsg(String s) {
            String errorMsg =(String)errors.get(s.trim());
      return (errorMsg == null) ? "":errorMsg;
 }

 public FormBean() {
  firstName="";
  lastName="";
  email="";
  userName="";
  password1="";
  password2="";
  zip="";
  notice="";
  errors = new Hashtable();
 }

 public String getFirstName() {
            return firstName;
 }

 public String getLastName() {
            return lastName;
 }

 public String getEmail() {
            return email;
 }

 public String getUserName() {
            return userName;
 }

 public String getPassword1() {
            return password1;
 }

 public String getPassword2() {
            return password2;
 }
```

```
public String getZip() {
        return zip;
}

public String getNotice() {
        return notify;
}


public String isRbSelected(String s) {
    return (notice.equals(s))? "checked" : "";
}

public void setFirstName(String fname) {
        firstName =fname;
}

public void setLastName(String lname) {
        lastName =lname;
}

public void setEmail(String eml) {
        email=eml;
}

public void setUserName(String u) {
        userName=u;
}

public void  setPassword1(String p1) {
        password1=p1;
}

public void  setPassword2(String p2) {
        password2=p2;
}
```

```
        public void setZip(String z) {
                zip=z;
        }


        public void setErrors(String key, String msg) {
                errors.put(key,msg);
        }

        public void setNotice(String n) {
                notice=n;
        }
}
```