Team DEC13-15

Task Manager

Project Plan

CONTENTS

TEAM COMPOSITION

SENIOR DESIGN, TEAM DEC13-15

CAMERON LEGLEITER, SENIOR-SOFTWARE ENGINEERING

DALIA ABO SHEASHA, SENIOR-SOFTWARE ENGINEERING

MARTIN STROBEL, SENIOR-SOFTWARE ENGINEERING

CLIENT/ADVISOR

SHASHI GADIA, PH.D

OVERVIEW

EXECUTIVE SUMMARY

Communicating information quickly and efficiently has become a high priority. From big businesses collaborating on new products, to small group projects trying to decide on what they should focus their work on, being able to have up to the minute information is paramount. The market for increasing ways of communicating information and viewing data easily is still growing today. With the help of our advisor, our Senior Design team will develop a functional prototype that can be used for gathering and viewing data, focusing on delivering a product that is easy to use and utilizes new technologies for providing a unique user experience.

Our product will be a website that allows users to create forms composed of individual fields. Initially fields will be simply two strings, one prompting a user for information, and the second the user's response.

For example, a college secretary may be tasked with collecting the next semester's list of textbooks. To do this, they must collect the ISBN for each textbook from each professor. In lieu of manually sending emails and collecting responses from, the secretary would setup a form on our website. The form they created might have the fields: course number, sections, ISBN. Once the form is created, an email will be sent to all of professors in the department. The professors, will open a link in the email, and respond to all prompts in the form, and submit it to a central database.

The Senior Design team will be made up of 3 Software Engineers from Iowa State University; we will research, design and implement a product that will meet and potentially exceed the basic requirements set out by our client.

The following document will outline the overall plan of creating and developing the solution.

DEFINITIONS

Schema	An XML schema as defined by the w3c. A more thorough definition of them may be found here: http://www.w3.org/standards/techs/xmlschema
User	A persons responding to a survey.

Survey	A collection of prompts, to which users should respond.
Administrator	A person who will be in charge of a set of records. They will be the ones to send invitations to take surveys, and the ones allowed to review the responses to those surveys.
Developer	A person that will be making purely technical changes, editing code, and doing other development tasks.
Administrator	A person who will perform maintenance tasks on the site.
HTML5	The most recent web standard for developing compliant websites. More information can be found here: http://www.w3schools.com/html/html5_intro.asp
AJAX	Asynchronous JavaScript and XML. Allows for content to be dynamically added or removed from a website.

PROBLEM STATEMENT

Existing solutions to collecting and managing data, such as Google Forms or Doodle, are relatively unsophisticated. Data can only be stored in a flat manner, such as spreadsheets, query tools are almost nonexistent, and permissions are very rudimentary. Additionally, gathering information from multiple individuals can be a daunting task, especially when the information needed is not in an elementary format. Along with this, when an administrator sends out a question to users, the users themselves may not be able to give a response quickly, which delays overall results from when the original question was asked.

GENERAL SOLUTION

Our solution is to create an HTML5-compliant website that utilizes an XML-based database for storing information. Administrators will be able to log in, create a survey with one or more questions, and submit the survey to multiple users who may respond at their convenience. If the users do not respond to the survey after a certain amount of time, the administrator may send reminder emails to all users who haven't responded to the survey.

After the administrator sends out the initial emails to all the respondents, s/he can view all of the responses to the questions asked in the survey. All of the results from the respondents will be stored in an XML Schema that is generated from the original survey. The administrator can also use XQuery-style queries on all of the data in the survey to view particular information rather than all of the data at once.

EXPECTED END PRODUCT

A fully functional website. This website should be easy to use (but ugly as dirt) and meet all of the requirements defined later in this document.

INTENDED USERS

For this application to be as useful as possible, we will attempt to appeal to as broad an audience as possible. We will assume that people who wish to use our product have a basic knowledge of working with websites.

The demographic we hope to reach is also quite broad. Essentially, we want to appeal to anyone that must collect data from people. Examples of people in these roles could be secretaries, club presidents, teachers, or people working as a team in a school project.

CONSTRAINTS

WEB BASED SOLUTION

Our solution must be a website, not a client application that users must install. This limits the languages and technologies that we can use for development, but allows for almost absolute platform independence.

XML

As our advisor is an expert in the field of XML storage, we will only be using XML based solutions for our project. For the most part, the hierarchical and extendable nature of XML will add much power and convenience. However, if this application is ever scaled to allow thousands of users to be working concurrently, it may create a performance bottleneck. In that case, we may have to consider creating a hybrid solution that utilizes other technologies such as SQL for administrative functionality.

FEASIBILITY

STORING RECORDS

Because of the decision to use XML for the primary storage tool, the task of designing the storage of complex data types becomes trivial.

CREATING FORMS

Graphically speaking, there are many entries in the market, where html forms are created and modified in real time. Using jQuery, this should be a relatively easy task, at least conceptually. An example of what our desired outlook will look like can be seen in Google Forms.

The next challenge in creating a form is taking an existing HTML form and converting it into a Schema. The trick will be either creating a data structure that can be communicated via a POST request, or writing a JavaScript library to handle that creation. Either way, it seems accomplishable to handle this. Ideally speaking, we would allow for the recursive definition of complex data types.

GENERATING RESPONSE FORMS

Going the other direction, we will certainly have to write a JavaScript library that interprets a Schema into an HTML form. Particular attention will have to be paid to the details of this, for example entry multiplicity.

MARKET RESEARCH

Beyond simply emailing our to all respondents a small email "survey," several solutions already exist for gathering information:

SIMILAR PRODUCTS

GOOGLE DRIVE FORMS

Allows users to easily create basic forms for free. In many respects, our solution will strive to emulate their success with this portion of the process.

Google Forms also works in conjunction with Google Analytics, allowing for powerful interpretation of large sets of data.

DOODLE.COM

In many ways this is little more than a cross platform calendar scheduling site. Doodle's primary functions is gathering information from a group of people based on a Boolean availability time grid. The results of these surveys can be added to different calendar applications, such as Google Calendar, Outlook, and others.

It is worth noting that one does not need to create an account to use Doodle, making it very convenient to include people that have never used it before. However, the interface is often described as clunky, limiting, or hard to use.

SURVEYMONKEY.COM

SurveyMonkey.com allows a survey to be composed of several pages of forms, allowing for users to neatly partition fields.

It also integrates with social media sites such as Facebook and Twitter, as a method of encouraging responses.

Their business model allows for small forms to be created for free, but requires payment for larger, more complicated forms.

ESURVEYSPRO.COM

Similar to SurveyMonkey.com, eSurveysPro.com allows users to create multiple pages for their enquiries. These surveys can be over anything from customer satisfaction to human resource questionnaires.

Their business model is more liberal, and allows unpaying customers unlimited surveys, questions, and responses.

OUR MARKET POSITION

What makes our solution unique is both in the way users can create complex form questions, and how users can retrieve data from the responses. Rather than having a large amount of statistics thrown at the curator, they can sift through the data using XQuery queries to view specific responses. Data overviews, such as overall counts of a specific response, can also be done.

With the end product being an HTML5 compliant website, we will have the functionality of HTML5, CSS3 and JavaScript to provide an easy-to-use and aesthetically pleasing experience to the users. We will also take advantage of AJAX technologies to provide dynamically-loaded content and smooth transitions.

RISK & MITIGATION

The following table shows major risks to the success of this project. We have calculated the probability, criticality, and mitigation strategy for each risk.

Risk	Mitigation Strategy
Difficulties with generating XML schemas from forms, and vice versa	As this is a large part of the system, most of our resources will be devoted towards ensuring dynamic schema generation works as flawlessly as possible.

	Since the forms will be in HTML, converting between HTML and XML should not pose too much difficulty.
People are reluctant to abandon their current form of collecting user information	We are already aware of many faculty on Iowa State's campus that would greatly benefit from utilizing a system like this. The system itself could be utilized for market research to determine whether or not it would be worthwhile to have.
The services needed to support the form generation/data gathering run too slowly to support a responsive UI	A majority of our design focus will be on creating an efficient client-server relationship. XQuery allows us to utilize a fast querying system with the database to minimize wait time for users.
Users produce more data than our application can support	XML is quite flexible in storing vast amounts of data and keeping it organized. Utilizing this functionality will allow us to handle larger amounts of data.

REQUIREMENTS

NON-FUNCTIONAL REQUIREMENTS

ID	Name	Description
1	Modular Security	Each user should be allowed specific access to specific fields. That access should also be able to be toggled independently on Read and Write levels.
2	Record Storage	All responses from users should be collected and kept indefinitely.
3	Scalability	The performance of our application should not be compromised by a large number of forms, users, or fields.
4	Easy to Administrate	It must be easy for non-technical administrators to look at all of the data that has been collected, create new forms, and remind participants that their responses are due.

5	Maintainable	It should be easy to improve our product by adding functionality, or fixing bugs that are found.
6	Responsive	Users should not be forced to wait for our application to process their requests.

FUNCTIONAL REQUIREMENTS

ID	Name	Description
1.1	Accounts	All people associated with our site will have a personal account.
1.1.1	Account Handle	Each account should have a unique string associated with it that identifies the individual owner of the account to other users.
1.1.2	Account Password	Each account will have a unique password that is used for authentication.
1.1.3	Form Creation	Any account will have the permission and ability to create a form.
2.1	XML Based	All data that has been captured from users should be stored in the form of XML based documents.
2.1.1	Document Content	Responses to forms should be stored in an XML documents corresponding to that document.
2.1.2	Document Layout	There shall be a Schema defined for each Form, that dictates what will be allowable content in the XML document containing all of the responses.
3.1	User Response Collection	Users shall be able to respond to surveys which they have been invited to respond to.
4.1	Response Template Generation	Given a Schema, a page must be able to convert that data into an HTML form for the user to complete.

4.1.1	Form Input	Most portions of the form will be “primitive” data types.
4.1.1.1	String Input	Forms should support basic varchar type input
4.1.1.2	Number Input	Forms should allow for the entry of integer or double types that are distinguishable from raw strings.
4.1.1.3	Date Input	Selecting a day from a calendar
4.1.1.4	Time Input	Selecting a moment in time on a clock, the smallest increment of time supported should be the second.
5.1	Form Product	The result of creating a new survey will be a Schema.

RESOURCES

Saxon	When it becomes necessary, our team will be granted a license for the Enter Edition of Saxon, an XML API for easy referencing in XML documents.
GitHub	In order to better collaborate between team members, we will be using a private git repository for all of our versioning.
TomCat	For development, each developer shall deploy our code using a local TomCat 7 service.