Team DEC13-15

Task Manager

Design Document

Dalia Abo Sheasha

Cameron Legleiter

Martin Strobel

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

INTRODUCTION

PURPOSE

This document specifies the design decisions made for our product. It will include the rationale for our decisions as well as alternatives considered in the process. The document will describe the front and back end of our product. It will describe the different modules of our software and how they interact with each other. The overall goal of the document is to allow any team to be able to replicate our project based on the information we provide.

DEFINITIONS

Schema	An XML schema as defined by the w3c. A more thorough definition of them may be found here: http://www.w3.org/XML/Schema#dev
Participant	A persons responding to a form.
Form	A collection of prompts, to which participants should respond.

Field	An entry within a form that a participant must respond to. Fields share a many-to-one relationship with forms.
Records	A collection of individual responses to a form dialog.
Administrator	A person who will be in charge of a set of records. They will be the ones to send invitations to complete a form, and the ones allowed to review the responses in the form records.
Developer	A person that will be making purely technical changes, editing code, and doing other development tasks.
Support	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.w3.org/TR/html5/
AJAX	Asynchronous JavaScript and XML. Allows for content to be dynamically added or removed from a website.
CRUD	Acronym for "create, retrieve, update, and delete". Used when defining basic functionality on databases or other forms of persistent data.
XPath	A technology developed to efficiently run queries on large XML documents.

EXECUTIVE SUMMARY

The product described in this document is an HTML5-compliant website that allows users to collect and manage data. This is done by letting administrators create forms containing fields that are later filled out by the recipients of the form. Our software will allow administrators to send out alerts when a participant has not filled out their form. After responses are collected, we will allow administrators to query specific information using XQuery-style queries.

OVERVIEW

Section 1, Introduction, includes a description of the overall project, definitions, and an executive summary.

Section 2, System Overview, provides a brief glimpse of basic architectural decisions.

Section 3, Development Standards, describes the stylistic decisions and tool-chain that the team used in development.

Section 4, Software Component Description, goes in detail about individual module responsibilities.

Section 5, Testing Specification, describes what needs to be tested across different portions of the project.

SYSTEM OVERVIEW

SYSTEM ARCHITECTURE

Our system will be based on a standard client-server website layout. Clients will connect to the website and interact with the pages on the front end, which will send requests to a Tomcat server backend and handle the requests. The front end websites will utilize the most recent HTML5 and CSS3 standards to provide up to date aesthetics and functionality.

DEVELOPMENT STANDARDS

DESIGN METHOD AND STANDARDS

We are using an object-oriented design approach with the back end of our system. This will allow for a more modular structure, and make new feature additions and code reuse much easier.

Most notably, the actual database implementation will be hidden behind a layer of abstraction. For our implementation, we were advised to focus on an XML approach. It would, in principle, be simple to write alternative implementations. Some alternatives may employ a relational-database such as an SQL server. Others may seek to gain efficiency by storing data in JSON, the way it is passed between the front and backend.

DOCUMENTATION STANDARDS

Each Java file created will contain a comment describing the purpose of the file and contained class(es) or interface(s). All methods within the previously mentioned files will also be commented with the description, arguments, and any return information. Any additional commenting within each file will be up to the developer's discretion.

The comments mentioned above will employ Javadoc, and HTML manual pages will be generated to make browsing the documentation more convenient.

NAMING CONVENTIONS

JAVA

Within the backend, which will employ Java, we will be following the Code Conventions for the Java Programming Language (Sun Microsystems, 1999).

HTML/CSS

In regard to front-end development, we will be following the Google HTML/CSS Style Guide as our primary method for standardizing the layout of our websites (Google, n.d.).

JAVASCRIPT

All JavaScript files will follow a format similar to Java naming conventions, using the Code Conventions for the JavaScript Programming Language

SOFTWARE DEVELOPMENT TOOLS

We will be using the Eclipse EE IDE for developing our JSP, HTML, CSS, JavaScript and Java programming. For version control, our team will use Git for keeping all members up to date on the most recent changes, with our remote repository hosted on GitHub to keep track of commit information and issues tracking.

OUTSTANDING ISSUES

OBJECT TO JSON MODULE

Currently, we are having the Java Objects create a JSON string and passing that as a response back to the Servlets. Our main concern is being able to convert XML attributes (information within a specific XML element tag) into a readable JSON format. More research will need to be done to see how this converting can be done in an efficient manner without too much overhead.

TESTING THE JAVA SERVLETS MODULE

Unit testing of modules that has proven verbose and error prone. In an effort to combat this, we must either develop or find libraries to ease our woes, or compensate with alternate forms of testing.

Given the effectiveness of manual testing in this case, we will likely use this as our exclusive manner of testing the servlets.

SOFTWARE COMPONENT DESCRIPTION

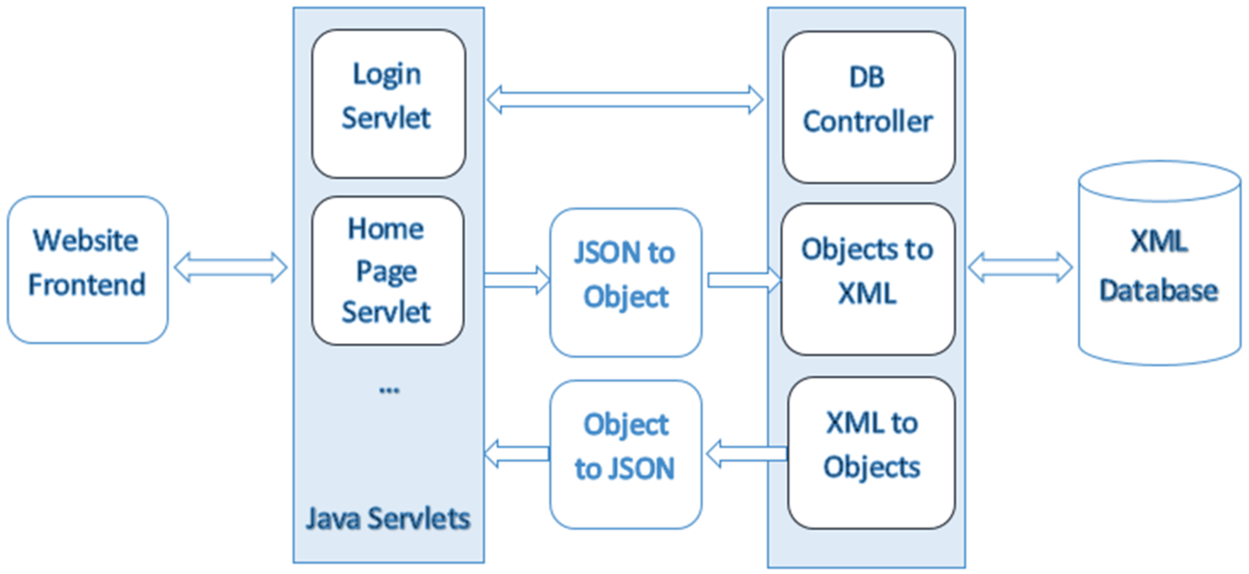

Figure 0.1 - Example of module implementations.

FRONTEND WEBSITE MODULE

SERVICE

The HTML5-compliant website that the administrators will use to create forms and the participants will use to populate responses to those forms. Administrators can view all forms that they have created and participants can view forms they are participating in and/or need to finish. An administrator will have the ability to decide whether responses are able to be reviewed by all participants, or only the one that submitted the response.

SECRET

How each webpage will interact with its corresponding servlet, including organization of data created by users and how that data is sent to the servlet (AJAX, data types, etc).

USES

Java Servlet Modules

JAVA SERVLET MODULES

SERVICE

Contains a multitude of servlets. Each servlet corresponds to a specific webpage on the front end, and handles the different GET and POST requests from its corresponding page. For example, the Login page corresponds to a Login Java servlet.

SECRET

How a servlet processes a request, how it interacts with the other backend modules.

USES

XML to Objects Module, Objects to XML Module.

HTML/POST TO OBJECTS MODULE

SERVICE

Deprecated

Used with form creation. Takes in HTML data that represents a form, and creates a series of Java objects to represent the form and information contained within.

After some experimentation with this module, we ultimately decided that this module was less effective than simply passing JSON between the servlets and front-end.

SECRET

How a form will be converted from HTML elements to Java Objects.

USES

DB Controller Module

OBJECTS TO XML MODULE

SERVICE

Converts a series of Java Objects into an XML Document that represents the initial HTML form.

SECRET

How Java Objects will be converted into an XML Document.

USES

XML to Objects Module

XML TO OBJECTS MODULE

SERVICE

Converts an XML Document into a series of Java Objects.

SECRET

How an XML Document is interpreted and instantiated into Java Objects.

USES

Objects to JSON Module, Objects to HTML Module

OBJECTS TO JSON MODULE

SERVICE

Converts a series of Java Objects into an easy to interpret JSON object. This module will mostly be used with AJAX calls to dynamically load/unload data into a webpage

SECRET

How Java Objects will be serialized into a JSON string.

USES

Java Servlet Module

OBJECTS TO HTML MODULE

SERVICE

Converts a series of Java Objects into HTML. This module will be used primarily on page loadings to help statically create webpage content.

SECRET

How Java Objects will be serialized into HTML

USES

Java Servlet Module

DB CONTROLLER MODULER

SERVICE

Delegates operations to the concrete implementation of a database module and conversion modules.

SECRET

Knows which database strategy is currently employed.

USES

XML Database Module

XML DATABASE MODULE

SERVICE

Performs basic CRUD operations, utilizing XML for storage, and emphasizing XPath for query operations.

SECRET

Interaction details with any database related technologies.

USES

XPath

TESTING SPECIFICATION

FRONTEND WEBSITE MODULE

TEST DESIGN SPECIFICATION

We need to ensure that the website as a whole can handle large amounts of requests simultaneously without being too slow or crashing completely, especially when a form has a large amount of participants and needs to handle updating multiple records on the fly.

FEATURES TO BE TESTED

The website's ability to create forms based on a big number of requests. Integration tests will be used to hand out multiple requests and confirm that the requests are handled properly by the module.

FEATURE PASS/FAIL CRITERIA

Our tests succeed if the website receives all the requests and creates the corresponding forms successfully. If there are any inconsistencies between the request sent and the form created, our test fails.

JAVA SERVLET MODULE

TEST DESIGN SPECIFICATION

While there are multiple servlets that each correspond to a specific page, the servlets must be able to take in specific requests, perform some algorithm or calculations, and return a response back to the page.

FEATURES TO BE TESTED

Checking to see if valid requests to a servlet return a valid response, and invalid or malformed requests return invalid responses or correct HTTP errors (e.g. accessing invalid or non-existent pages return a 404 server error).

Some invalid requests should return valid responses. Handing off incorrect login credentials to the Login servlet should return a valid “Incorrect login credentials” response.

FEATURE PASS/FAIL CRITERIA

If a valid request is handed to a servlet (i.e. correct login credentials are passed to the Login servlet), then a valid response should be handed back (user should be redirected to the index page). If an invalid or malformed request is sent, then an invalid response should be given back.

OBJECTS TO XML MODULE

TEST DESIGN SPECIFICATION

Each element in an HTML form will have been converted into a respective Java Object. Each object should also have respective methods for outputting both XML Schema information (in an XSD file) and XML when called.

FEATURES TO BE TESTED

For each Java Object created, we will test if the XML outputted matches the respective HTML it was initially converted to.

FEATURE PASS/FAIL CRITERIA

Tests will pass if the objects outputted XML matches our expected. If the output does not match expected values, then this would be classified as a failure.

XML TO OBJECTS MODULE

TEST DESIGN SPECIFICATION

The XML Schema information in the XSD file that comes into this module will need to be converted into a java object that conforms to the XML Schema.

FEATURES TO BE TESTED

For each XML Schema file generated, we will test if the object it is converted into has the properties specified by the XML Schema.

FEATURE PASS/FAIL CRITERIA

This module's tests will be combined with 6.4 module's test which handles the opposite conversion from objects to XML Schema. The java object handed in to module 6.4 is converted to XML Schema and is converted back to an object by this module. If the object the modules started out with match with the object handed back by this module, then our tests pass. Otherwise, if there are any discrepancies, our algorithms would have failed somewhere in the process.

OBJECTS TO JSON MODULE

TEST DESIGN SPECIFICATION

When this module is handed a Java Object, it should convert it into a JSON string. It is important that this module converts the data correctly for handing back to JavaScript.

FEATURES TO BE TESTED

The ability to take in a Java Object and convert it to JSON data that has the same properties.

FEATURE PASS/FAIL CRITERIA

Tests from this module will check that an object correctly returns a JSON object with all properties and attributes included in a readable manner. Improperly formatted JSON will be considered a failing test.

XML DATABASE MODULE

TEST DESIGN SPECIFICATION

As this module is the main entry point to the database, extensive testing will need to be done to ensure overall functionality.

FEATURES TO BE TESTED

Standard CRUD functionality.

Creating a form adds the correct information to the database, as well as creates the associated files necessary and references them in the database.

Querying to get a form returns all the necessary objects from the XML, and nothing more.

Storing participant's responses in the correct locations, and retrieving them.

FEATURE PASS/FAIL CRITERIA

Any issues related to an inability to find the database files or malformed return results are considered failing. Returning too much or too little data than expected is also a failure. Returning equivalent data as expected, and locating correct files would be passing.

STUB DATABASE MODULE

TEST DESIGN SPECIFICATION

This module will be used only for the sake of testing other modules, and will be verified through unit tests and validated through use with the rest of the application.

FEATURES TO BE TESTED

Standard CRUD functionality.

Limited ability to query information.

Ensured lack of persistence, the data should be the same every time the application is launched.

APPENDIX A - SERIALIZATION STRATEGIES

GETFORM

A reference for how

REQUEST

Format	**REQUIRED Type:** enum **Options:** “HTML”, “JSON”, “XML” Dictates what format the data you receive will be in.
ID	**REQUIRED Type:** string The unique identifier of the form that you would like to fetch
Title	**Optional Type:** bool If this option is true, the title of the form will be returned. Otherwise, the response will not include such an entry.
Description	**Optional Type:** bool If this option is true, the description of the form will be returned. Otherwise, the response will not include such an entry.
Questions	**Optional Type:** bool If this option is true, the description of the form will be returned in the response.

REFERENCES

Google. (n.d.). *Google HTML/CSS Style Guide*, 2.23. Retrieved from Google Code: http://google-styleguide.googlecode.com/svn/trunk/htmlcssguide.xml

Sun Microsystems. (1999, April 20). *Code Conventions for the Java Programming Language*. Retrieved from Oracle: http://www.oracle.com/technetwork/java/codeconv-138413.html

W3 Consortium. (2012, June 22). *XML Schema*. Retrieved from W3: http://www.w3.org/XML/Schema#dev

W3 Consortium. (2013, August 6). *HTML5*. Retrieved from W3: http://www.w3.org/TR/html5/

DOCUMENT CONTROL

Title:	Design Document
Issue:	Issue 2, Draft 1
Date:	December 11, 2013
Author:	Team Xmlers
Distribution:	Online
Filename:	DesignDocument.pdf

DOCUMENT CHANGE RECORD

Date	**Version**	**Author**	**Change Details**
December 11, 2013	Issue 2 Draft 1	Team Xmlers	Updating module usage.