LetsGITit - Implementation Document

Table of Contents

1. Project Title and Authors

1.1. Assigned Team Number

1.2. Team Name

LetsGITit

1.3. Team Members

Name	USC ID	USC Email
Edward Han	7421483867	hane@usc.edu
Joyce Wang	7039608014	joycew@usc.edu
Robert Diersing	3445624516	rdiersin@usc.edu
Summer Seo	8321974769	summerys@usc.edu
Woonghee Lee	5094791538	woonghel@usc.edu

2. Preface

This document will preview some implementation and design changes of the KnowItAll web application. It outlines some of the necessary changes that are required in response to added requirements that are required of KnowItAll. These changes in architectural and detailed design, and requirements reflect the need to add a social component to KnowItAll to enhance searching functionality.

3. Introduction

KnowItAll will be implemented with some key changes to our initial design and functionality. Initially we had planned to use java servlets and jsp for our backend functionality, but have all agreed to use javascript to implement the entirety of our web application. The following explanations below detail our rationale for these changes compared to our initial design proposal.

4. Architectural Change

4.1. What architectural changes did you make during implementation?

There were no noteworthy changes to our main architecture. However, we did expand on our existing architecture based on the different frameworks that we decided to use. The new frameworks that we decided to partake in maintained the overall integrity of our three-tier architectural design.

4.2. What's the rationale behind the decision change?

The most significant changes were the following: programming language (java to javascript), framework change (java servlets and jsps to AngularJS controllers for server, ReactJS to AngularJS for client-side functionality, and removing JDBC). However, all these changes did not compromise the entirety of our architectural structure. The changes made simply replaced our approach to our detailed design, but they still adhered to the three-tier architectural design pattern that we initially proposed. We found javascript to be less superfluous and more concise than java, and because we made this switch, we naturally incorporated existing frameworks to our project like AngularJS and ExpressJS. Although AngularJS was used and should be used for client-side functionality, controllers, which is an AngularJS-specific functionality allowed us to use AngularJS for a small portion of server-side implementation and mostly client-side data input processing. Our simple server was set up with a Node.JS server and we are continuing to use the same database, MySQL. The controllers also effectively queried the database and provided a fast connection between the client and database through our simple server.

5. Detailed Design Change

5.1. What detailed design changes did you make during implementation?

Our original design was focused on using java servlets and jsp's to handle data transfer and processing. However we ultimately decided that we are changing our design and implementation to using Javascript. We will be making use of the vast open-source JS resources online, such as AngularJS, ExpressJS, and NodeJS.

The use of AngularJS controllers was a major design change that we implemented. We defined an AngularJS module that acts as the application that can serve different requests from the client. Our controllers were able to serve and process these requests, and a controller was assigned to each of our separate html pages that represented the different page links on our navigation bar.

For instance, we designed our UI to be a single page application, and to implement this we used Angular JS to connect each of our different “page” contents to a controller. The controller acted as the intermediary glue between the client and server, as well as indirectly between the client and the database. The controller effectively performed this job by extracting input data from user input in a form/text field and forwarding this data to the server, so the server can query the database accordingly.

Our NodeJS server maps itself to many different controllers, where each controller is mapped to one html page. With this mapping, our server handles client requests easily by the corresponding input id. Therefore, as long as the controller process client data appropriately, the server simply queries the database with an sql statement.

The queries are supported by a mySQL connection that the server creates and by ExpressJS, which improves connection speed for sending requests to the database and fetching relevant information to be displayed back to the client, at a significantly faster rate.

We made some changes in our class diagram and database to add more functionality. We are adding two tables in our sql database, UserToFollowing and UserToFollower, in order to keep track of the current user’s followers and the users the current user is following.

This list of users can be used as one of the ways to decide more relevant content to display to the current user.

Also, to add the ability for users to like/dislike the polls/ratings, each survey will have a counter for each like and dislike and a table that holds an integer key and a bool value (where int refers to the student id and bool refers to whether that student liked/disliked ever) to keep track of who liked/disliked the post. Using the table, we will determine the status of the user on a specific survey (whether the user clicks like, dislike, or hasn't clicked either (none)) and perform an action according to that.

5.2. What's the rationale behind the decision change?

Javascript will consolidate all of our web-based functionality into one simple format that we feel is more concise and simple than dividing our connections between clients, server, and database, into java servlets and jsp's. Using javascript will give us more immediate access to powerful libraries and API's that are crucial to our UI design. We initially felt limited with the idea of java servlets and jsp's, since javascript has more access to powerful online libraries and frameworks, and java servlets and jsps didn't seem to offer any well-known extensions.

The concept of controllers provided to us in AngularJS was extremely useful for client-side data processing, and it accomplished this by strictly separating the logic from the user interface and the server. This provides many advantages as it abstracts away all the work that either the client or server would need to do in, say if we continued to use java servlets and jsps. All processing happens only in the controller and it makes it easier to understand what each file in our project is doing, instead of debating back and forth between if the client and server are both participating in data processing. Consequently, our server became much more refined and simpler, and simply calls functions defined in the controllers. And lastly, since we aren't using java anymore, there is no utility for JDBC, and thus we have decided to remove it from our implementation.

Although java servlets and jsp's aren't an invalid way to approach web development, our team felt that building off of javascript with various frameworks was the most clear, modern, and proper way to approach web development, and reflects our understanding of modern practices and principles used in the software engineering/web development industry.

6. Requirement Change

6.1. Does this change our design?

There were no significant changes to our requirements design. While the architectural design and architectural style stayed the same, the biggest change from the new requirements is the search function that counts number of tags, and number of likes

and dislikes.

6.2. If so, what should be changed and how?

We created new database tables that hold the number of likes and dislikes for each corresponding poll or rating. Also, the user table now holds following users' userID.