Implementation

Crowd-sourced Rating Application

Team Byters

Emerson Ye

Matthew Sternquist

Rishabh Gupta

Medhavi Joshi

Weston Newby

Revision 1.0

TABLE OF CONTENTS

Revision History

Date	Description	Revision	Editor
03/28/2018	Document Creation	1.0	Medhavi Joshi
04/05/2018	Document Modification	2.0	Medhavi Joshi
04/07/2018	Document Modification	3.0	Rishabh Gupta

1. Project Title and Authors

a. Crowd-Sourced Rating Application

b. Team Byters

i. Emerson Ye 010633090
ii. Matthew Sternquist 011039353
iii. Rishabh Gupta 010013666
iv. Medhavi Joshi 009980334
v. Weston Newby 009100078

2. Preface

This implementation document describes the implementation of the crowd-sourced rating application. This project is a rating application for students with valid SJSU IDs in order to create polls and/or write reviews anonymously and rate questions. This could be used for food, professors, classes or anything else.

3. Introduction

The purpose of this implementation document is to provide a description of the various changes that were made when looking back on the system design . These changes come from

implications or potential requirement changes/adding new features that were not initially within the design scope.

4. Architectural Change

A. What architectural changes did you make during implementation?
=> We sticked to the design pattern we decided to use, Model-View-Controller. We have a 'Connect' class that played the role of the controller to communicate between the Model and the View.

B. What's the rationale behind the decision change?
=> We did not make changes to the architectural design.

5. Detailed Design Change

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

1. We changed the name of many classes from what we defined before. We went over the article "Best Practices to follow while coding." This gave us a better understanding of how we should name the classes and the variables.
2. We did not make the classes: user, poll, rating, pollList, userSubmissionList. Instead, we made PHP URL's and extracted the information directly from the database through the Apache server. And then displayed them in the ListView on the activities: viewAllPolls, viewAllRatings, viewAndEditRating, viewAndEditPolls.
3. For Registering/Signing up a new user, we are not implementing the salting and hashing of password for now.
4. For creating a new rating, we use the Connect class method that uses http POST to send the information to the aws server using php and inserts the data on to the database.
5. For creating a new poll, we use the Connect class method that uses http POST to send the information to the aws server using php and inserts the data on to the database. But here we have a different table for options and therefore we have another method that deals with creating dynamic options for the poll. The specific poll is handled using the specific poll id using MySQL query.
6. For voting for ratings, the request is generated through the Connect class method and then sent to the php server where the database is updated with the useful votes using MySQL query.
7. For deleting a poll, the Connect class method deletePoll takes in the rating title and sjsu id of the user then JSON parser object uses htttp POST to send it to the php server. The server then deletes that specific rating from the 'Ratings' database using the primary key:sjsu id and matching the rating title using MySQL query.

8. Deleting a user is done using the deleteUser method of the connect class that sends the sjsu id of the user through http POST to the AWS php server and then the MySQL query find the specific user through the user id and deletes that entry from the database.
9. To load a rating, Connect class object method getSpecificRating interacts witht he AWS server by passing rating title and sjsu id of the user, and the php interacts with the database to find that entry and responds to the client side with the array containing the entry. The Client side class UserRatings.java shows this specific rating to the client.
10. To loadAllRatings, it is very similar to getSpecificRatings method from connect class. Just that here we don't have to pass in any parameters. We directly interact with the php server and the server interacts with the database to load all the ratings. The Client side class AllRatings.java helps display the ratings in the proper format.
11. To load all polls, we use loadAllPolls.java method from the connect class and the php server loads all the polls from the server and displays it to the client side using AllPolls.java class.

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

We are using AWS server to use RDBMS (relational database management system, SQL). For the backend, we used Apache instead of javascript. The reason to switch to Apache is because the team knew how to use PHP and Apache already. So it helped us implement better and more efficiently in the time we had for the project.

6. Requirement Change

a. Does this change your design? If not, why not?

Yes, it changes our design. The main design change occured because we chose to use AWS server using php scripts to interact with Apache for our backend instead of Node JS. In the current design, Connect class connects the client side to the server side.

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

While creating new polls, there should be a dynamic option creation available to them. We are using two different tables for this purpose and each entry could be accessed using the sjsu id and poll title. The database tables must be joined using the MySQL query to do so in the php script.

c. Has this new requirement been implemented? Or will be implemented in the future?

This new requirement is not implemented as of now but will be implemented in future. We are planning to apply the changes before the testing.