Anna Nardelli

Dr. Lakshmanan

CS 517

18 December 2023

Final Paper

Introduction

The concept for the final project for CS 517: Engineering Web-Based Systems is based on a survey application. The project is intended to allow users to create, view, and respond to surveys. This topic allows for the application of many different programming concepts and languages, including MVC structure, php, HTML, CSS, mySQL, phpMyAdmin, and JavaScript. The final result of this project is to be a complete, functional full-stack web application. The website requires the creation of an account in order to access the content. Each user must be logged in to view, create, or respond to a survey. The user will only be able to view the responses for the surveys they have created. The model-view-controller (MVC) application structure is used to handle the functions of the web application itself. phpMyAdmin hosts the database and php's PDO library is used to interact with the database. The views of the project are constructed using HTML to display the structure of each page in an organized manner. The final implementation of the project is stylized using CSS to make the website more aesthetically pleasing. JavaScript is used to validate forms before submission so that the database does not become corrupted with bad data that does not follow the intended structure of the form submissions. The combination of all of these concepts creates a complex and functional web application that can continue to be improved upon to eventually produce a high-quality and reliable service to its users.

Implementation

The ER model shows the structure of the database in a visual format. There are six tables in the database, including login, survey, question, choice, survey_response, and survey_answer. The login table stores information about the user. This includes first name, last name, email, and password. Each user also gets a unique ID number to identify them by even if there are two users of the same name. The password is hashed on the database side using SQL's PASSWORD() function. When viewing passwords in the database, it appears to be a random string of characters so that those with access to the database cannot access a user's password directly. The survey table stores information about an individual survey. This includes the name of the survey, the ID of the creator of the survey, and a unique ID number to identify the survey itself. The question table stores information about individual questions from surveys. It includes a unique ID number for each question, the ID number of the survey it is associated with, the type of question, the text of the question, and an integer denoting whether the question is required. The choice table stores information about possible choices for multiple choice question types. It includes a unique ID number for each choice, the ID number of the question it is associated with, the text of the choice, and the order the choice should be displayed in on the survey form. The survey_response table stores information about a complete response to a survey. It includes a unique ID number for the response, the ID number of the survey it is responding to, the amount of time taken to complete the survey, and the ID number of the user who created the response. The survey_answer table stores information about the answer to a single question in a survey. It includes a unique ID number for each survey answer, the ID number of the survey response it is associated with, the ID number of the question it is answering, and the text of the answer. These

tables are implemented in the phpMyAdmin server and they serve as the database for the web application.

The web application allows the user to interact with the data in the database using the MVC structure. There are models for credentials, surveys, responses, choices, and questions. Each of the models access the database using php's PDO library. The credentials model validates users' login information and inputs new user data into the database. The survey model retrieves the surveys already created and submits survey responses. There are two functions for the retrieval of surveys: one that retrieves all surveys, and another that retrieves only the surveys owned by the ID number passed to the function. This is so that the latter method can be called when a user wants to view the responses of their surveys without retrieving information about the surveys they do not own. The responses model retrieves survey responses from the database. The choices model retrieves question choice options from the database. The questions model retrieves information about the questions of a specific survey from the database.

The controllers of the program use the methods in the models to retrieve the data from the tables needed to complete their functions. Some of these controllers only need to use one model and others need to use multiple. The surveyresponses and survey controllers utilize the GET request method to pass a survey_id value to the model from the URL. This allows the models to query the database for only the survey_id currently needed. The surveyresponses, login, newaccount, and survey controllers also utilize the POST request method to upload data from the user into the database. The controllers then include the view files to display the data retrieved from the models.

The view files from the program contain the HTML and CSS code that displays the pages. The CSS is implemented in the head regions of these files and the main HTML content is

towards the bottom. php is used in these files to display information from the database and HTML is used to structure that text. The CSS is from a template from W3's website, which is cited throughout the system. The views use JavaScript files to validate inputs before they are inserted into the programming.

The JavaScript files perform basic validation of the input data in an effort to reduce potential database breaches and invalid, messy inputs. If a user fails to fill in proper fields, a pop-up box appears telling them which fields are incorrect. However, if the fields are completed correctly, there is no pop-up box and the data is passed into the program without error. The JavaScript portion of the project creates a more secure and reliable program. This has been implemented for the create account, login, and create new survey post forms.

Results/Conclusion

The completed project runs all functions smoothly. The application does not crash, produce server errors, or show blank pages anywhere. The application adds data from the post forms to the database without error. However, time constraints meant that the implementation of the different types of questions could not be completed. The only type of question that the program is able to store is text. Future versions of the survey builder project will include checkbox and radio question types and implementation of the Choice table in the database. Additionally, the CSS is not completely cohesive, as the new account and new survey pages look very different than the other pages. This is because the W3 template did not include stylized post forms, so another W3 template was used to create the input forms on those two pages. Future versions of the project will include more cohesive CSS styling. Aside from the aforementioned shortcomings of the program, the project runs without error and accomplishes the major goals set out to achieve.