Sean Gor, Brennan Duff, and Kenny Chau

Database Management Systems

9 December 2024

Final Project Written Report

For our final project, we implemented a three-tier client server program. Similar to what we learned in class, this project modeled the process of the http protocol, such as posting, checking and retrieving requests via servers. After much contemplation, we decided to implement an online tutoring system called AnyTimeTutoring, analogous to the websites serving the Writing Center and Student Success Center tutoring programs here at Saint Joseph's. Originally, we were going to implement a payment system; however, to make it more similar to the idea of the Saint Joseph's tutoring programs, we decided to make all tutoring bookings free. Some of the main features included an about page, login and signup pages, and the ability to book, search for, and cancel an appointment. This idea was inspired by the Writing Center and Peer Tutor websites of our university. The backend server was made in Node.Js, and the layout of all pages was done in HTML and CSS. All of the data was made and organized into tables using SQL.

One of the main parts of the website was the login page. Upon entrance to the website, users are navigated to a login page asking them for a username and password. If they have not logged in before, there is an option to sign up for a new account (discussed later). This login page takes a user entered email and password and checks them with the database using a sql query. If the query is successful and the user exists in the database, the user is signed in and a session is started for the account. Otherwise, the user is not allowed in and an error message is

displayed. This is how session tracking is implemented. Without the session, the page automatically redirects to the login screen.

In addition, we implemented a signup page in case a user wants to create a new account (an account is required for booking tutors). When the sign up button is pressed, the user is taken to a new page The signup page takes a user-entered name, email, and password. The email is checked to verify that it is in the correct email format, and then all of the user information is inserted into the database using a sql query. If the email is not in the correct format, an error message is displayed. In addition, if the passwords (entered two times, once more to confirm), an error message is displayed saying, “passwords do not match.” Upon success, the user is redirected to the login page, with his/her credentials saved in the accounts table made in SQL.

After a user successfully logs in, he/she is directed to the main webpage. On this page, we have our Website title, AnyTimeTutoring, along with a welcome message. Near the bottom, we have a brief description of what our company stands for and our mission statement. HTML and CSS were used to change the font, font size, and make the text more visually appealing. We also included a picture of a SJU building to give the webpage more of a homey feel.

Next, a tutor page was created to show all tutors which were available. To make searches successful, a get method was made that retrieved data from the database as a JSON file and then displayed the data through an html file. A search bar was created to search up tutors by name and subject. A select statement from SQL checked if the user’s conditions matched the attribute values in the tutors table. If there were no tutors which matched the user’s conditions, a message was displayed indicating that there were no tutors available with the user’s desired information.

If there are matches, all tutors which match the user's conditions are shown to the user. The search process was made so that as the user was typing, the list of tutors was adjusted automatically based on the string typed in so far (not just at the end).

Last but not least, a page with a list of appointments was created to give users the ability to search for, book appointments, and cancel already booked sessions. As for searching for appointments, a get method was created which retrieved data from the database and turned it into a JSON file. Afterwards, a post method that displays the JSON data as an html file could be displayed. Under the appointments section, a user could look at any of his/her booked appointments (if any existed). This was achieved by a select statement which fetched all data in the appointments table for that specific student. In other words, appointments could only be shown if the student was logged in with a valid account. Furthermore, there was an option to book an appointment (shown on the tutors page). Users select the tutor and subject they want to be tutored in, and they input the desired date of the tutoring session. The dates were made such that a user could not book an appointment during days which already passed. The data is inserted into the appointments table in SQL, and when the user refreshes the webpage, that appointment will be displayed along with any other booked appointments.

In addition, let us suppose a student needs to cancel his/her appointment. They can easily do so by pressing the cancel appointment button on the appointment they wish to cancel. If a user cancels an existing appointment successfully, a success message appears. If not, an error message is displayed saying that there was an error canceling the appointment. When an appointment is

cancelled, it is removed from the appointments list entirely, so when a user refreshes the page, that appointment no longer shows up.

To facilitate transition of different components of the webpage, we designed a navigation bar consisting of the titles of different webpages. Like most other websites, it is shown near the top (but below the header) for users to seamlessly navigate between different pages. For example, if someone was on the homepage and wanted to see the list of tutors, all he/she would have to do is click on the “view tutors” button. This bar was designed neatly with html and css (along with the other webpages) to make it more aesthetically pleasing.