Julio Munoz, Jovan Golding, Maodo Seck

11/14/2024

Group Project Documentation File

For this group project, we have chosen to develop a system for a range of subscription management information. We will manage subscription services with renewal reminders and will create a system for payment processing. Our tools for this project would be MySQL in order to create the tables and the database, we will build our application using PHP for server-side processing, and HTML/CSS for the front-end. Jovan Golding would write the SQL portion of the project also design the system with CSS and wrote code to gather information from the user. He also created the user guide. Maodo Seck would write the PHP code, and I, Julio Munoz, would create the documentation part and handle submission of the project. We will work as a team to develop aspects of a subscription management service and provide information that is clear and easily understandable so that the majority of people can use the service.

I will begin by describing the content in the MySQL database. In the database, Jovan created tables for users, subscriptions, and payments. In the users table it contains information regarding the user id, username, email, user password, and a timestamp of when it was created. The user id is a primary key. In the subscriptions table there is data for the subscription id, user id, service name, start date and end date. It also states the status of the subscription whether it is active, inactive, or expired. The default status is ‘inactive’. In the final line of the table, it references the user id as a foreign key. The third table is a payments table that contains payment id, user id, subscription id, the amount of the payment, and a payment date. It contains the status

of the subscription whether it is paid or unpaid. The default status id ‘paid’. The foreign keys of user id and subscription id are referenced. These are all the tables needed for our project and to view them you just execute the last line in MySQL.

To access the subscription management system, you can start by opening the subscribe form HTML file. This will bring up text boxes that the user can fill out entering the information required. This data includes a username, an email address, password, service name, duration, and number of subscriptions. This then sends the information to the subscribe PHP file and gets processed. There is also a CSS file added to design the HTML page to our liking. The code written in the fetch subscription PHP file will connect to the database and get subscriptions and user data. If the information is valid, then it is passed on. In the renewal PHP code, it finds subscriptions expiring in the next 7 days. After doing so it sends a reminder email to the users saying to renew their subscriptions. A message echoes say that the reminders were sent. All of this was done by Jovan.

The following description of our project is the multiple PHP codes that were created by my teammate Maodo. There are eight total PHP scripts. They contain code to cancel a subscription, create a subscription, connect to the database, record a payment, register a user, renew a subscription, retrieve a user subscription, and to verify if a subscription is valid. All these codes work together to give us the information we are looking for. I will start by describing the cancellation of a subscription code. In this code there are various functions used. A cancel subscription function is created with the parameter of subscription id. Then a function is called to retrieve the database that you are going to work with. A try and catch statement is created in order to execute the exceptions being passed. The try statement begins with a begin transaction function. Then a SQL variable is created in order to update the subscriptions to ‘cancelled’. If the

statement matches the get transaction variable, then ‘subscription cancelled successfully!’ will be echoed. There is an exception in the catch segment of the code. If the transaction matches the rollback function, then ‘failed to cancel subscription’ is echoed. The rollback function acts as a safety to prevent partial or incorrect data from being persisted. That is the gist of the cancel subscription PHP code.

Next is the create subscription PHP. After the HTML code, there is a require block that will grab the database connection PHP code and send it over to the create subscription PHP code. Then there is an if statement where if the information matches, it will trigger the try and catch statement. In the try section, it connects to the MySQL database by calling the ‘getDBconnection()’ function. The next batch of coding is basically inserting and transferring information to match what is inputted. After the information is ran, it will echo a statement of ‘Subscription created successfully!’. If the information does not match, then the catch block will be executed and it will echo ‘Error creating subscription’. That is the subscription creation section of our project.

The following portion of code will be the database connection PHP code. This code will connect us with the MySQL database and transfer the information over. It starts with creating a function and variables within the function that contain information from the database. Another try and catch statement is created which takes the host, database name, username, and password from PHP and returns it to the MySQL code. All of this is done securely with the PHP data objects. If there is an exception, then the catch statement will execute echoing ‘Connection Failed’. That is how database connection PHP runs.

The next section is the record payment PHP script. It uses ‘require’ to snatch information from the database connection PHP. After that, it has an IF statement which verifies if the

information from the matches what is needed. Then it takes the subscription id and the amount from the PHP code and writes it to the MySQL database. In the try and catch block of record payment PHP, it inserts information to the database such as the subscription id, payment date, the amount, and status. It then executes the subscription id and the amount. If there are no errors, then it echoes ‘Payment recorded successfully!’. If there is an error, then the catch statement is executed and echoes ‘Error recording payment’ followed by the message. That is all there is to that section.

The next script is the register user PHP. It again uses the require statement to get data from the database connection PHP. In the IF statement, if the information matches ‘Post’, then the name and email will be sent over. The try and catch statement in this script deals with only the name and email values. It executes those values from the database and if it all goes well it echoes ‘User registered successfully!’. If an error occurs, then it echoes ‘Error registering user’ followed by a message. This was a simple code used to register a new client to the database.

Next up we have a renew subscription PHP. Here we create a function called renew subscription with the parameters of subscription id and amount. In that function we assign another function called ‘getDBconnection()’ to the variable $pdo. This allows us to connect with the database. In the try statement that follows, a begin transaction function is called. In this transaction, it updates the subscription id to a new end date. Then it prepares and executes that instruction to the database. The subscription id, payment date, and amount are inserted into the payments table. After that is executed, the code will echo ‘Subscription renewed successfully!’. If there is something wrong in the catch statement, it will echo ‘Failed to renew subscription’ with a message.

In the user subscription PHP, a function called get user subscription is called with the parameter user id. In this function, the getDBconnection() function is called and assigned to $pdo. This is where JOIN comes in. The $sql variable selects username from the users table. Also start date, end date and status from the subscriptions table. Then it joins the information from the users table to the subscriptions table on users and subscriptions from user id. A statement is prepared and executed on the user id. A subscription variable is made being assigned a statement and the fetch function. The subscription variable is returned and that is the end of the user subscription PHP.

The last code that I will go over is the valid subscription PHP. This code basically determines whether a subscription that is passed onto it is active, inactive, or expired. It starts by creating a function called isSubscriptionValid() with the parameter of subscription id. In that function, it connects to the database. Then it uses the select, from, and where clauses to retrieve information from the database. A statement is prepared and executed under the $stmt variable. A subscription variable is made having the $stmt variable fetched. In the If statement that follows, if all statements value equal ‘active’ and if the end date of the subscription is further out than the current date, then it will return ‘true’. Otherwise, it will return ‘false’. This is the final PHP code that was made and I have described the components of the codes and what they accomplish.

Working on this project helped me gain a better understanding of PHP code along with the use of MySQL database. This team divided the workload into three parts where we can contribute our own ideas based on the group project guidelines given to us by the professor. I hope this documentation serves well as a guide to our PHP code and will help describe what we have created. The process of integrating PHP and MySQL is a useful skill to gain to traverse

information and modify it. I think everything went smoothly in working with my other teammates to create this project.