Project Report

The goal of this project was to create a database-driven website that allows Aston students to book events. As the coursework specification suggests, I decided to only create 3 events that can be seen on the home page. Therefore, as we needed to list at least six events, each event is listed twice.

The PHP code for the home page of the website is in the “index.php” file, and all of the CSS styling for that page is included in the “styles.css” file. On the home page, as you scroll down you will see the events are initially listed according to their categories. There is a dropdown menu that allows you to order the events according to their categories or by their dates. I used Bootstrap cards for listing the events, and they contain an image, the name of the event, a brief description and the event's date. The user can click on the “More Details” button to view additional information about the event.

Additionally, I also used Bootstrap for creating the navbar and the footer which are both responsive to the users' screen size. I used PHP to change the content of the navbar when the user is signed in or not. For example, when the user is signed in, they can view the “My Bookings” and the “Sign out” links. However, when the user is not signed in, they are shown the “Register” and the “Sign-in” links in the navbar. As the footer and navbar needed to be on all the pages, I decided to create separate pages for both. The code for the navbar and the footer is in the “header.php” and “footer.php” files, respectively.

Moreover, the code for the registration and sign in page is included in the “register.php” and “signin.php” files, respectively, and the code for styling both pages is in the “userdetails.css” file. Both pages have HTML validation to make sure the forms have been filled out correctly. As well as the front-end validation, I have also included back-end validation using PHP. This is because the front-end validation provides a better user experience, but it can be bypassed easily. However, the back-end validation will make sure the details the user has entered are valid. Once the user has successfully signed in, they are relocated to the home page where they can browse the events. For the register page, there is also a JavaScript file called “validatePassword.js” that checks if both passwords are the same and long enough. Once the user has registered, they are taken to a separate page called “registered.php” which shows them an appropriate message to improve the user experience. The CSS code for that page is included in the “message.css” file.

Once the user clicks the “sign-out” button, they are taken to a separate page where they are shown a message to confirm they have successfully managed to sign out. The page consists of two files, “signout.php” and “signout.css”.

Each event has its own PHP page where the user can see additional information about the event, and like and book the event. For example, the event “5-a-side Football Tournament”, has its PHP code in the “football.php” file. The “Art Exhibition “and the “Live Talk” events have their code in the “art.php” and “tak.php” files, respectively. The CSS code for all the event’s pages is included in the “events.css” file. The students can show their interest in the event by clicking the “like” button, which also shows the total number of likes for the event. The “like” button also has a separate JavaScript file called “likeBtn.js”, which changes the colour of the button once it is clicked. The user must be logged in to successfully like the event, as the data is stored in the “likes” table in the database. Once they have liked the event, they are relocated to the “liked.php” file where their like is confirmed.

If the user has already signed in, they can book the event by clicking the “Book Now” button. Otherwise, they are relocated to the Sign-in page, as they need to have signed in before booking any event. The effect on the “Book Now” button is inspired by a YouTube video (Brian Design, 2020). If the user has successfully managed to book onto an event, they are relocated to the “booked.php” page where they are shown an appropriate message to confirm their booking. The CSS code for the “booked.php” page is in the “message.css” file.

As an additional feature, I have also made a “My bookings” page where the user can view all the events they have booked. The page consists of two files, “bookings.php” and “bookings.css”. Here, I also used Bootstrap cards to list the events. If the user clicks on the “My bookings” page when they have not signed in, they are redirected to the sign-in page.

Another challenging feature I have included is made the website responsive to the users' screen size. For example, the two events are shown side by side on a large screen, but for smaller screens only 1 event is shown, as the second event is then shown below the first event. I also used Bootstrap to make the navbar and the footer responsive to the users' screen size, as mentioned above. Also, for displaying the extra information on the events page, I used the “@media screen” rule to refactor how the information is displayed on smaller screens (e.g., @media screen and (max-width: 768px) for tablets and iPads). The code for that can be found in the “events.css” file.

You will notice that every time I insert any data into the database, I have used PDO prepared statements to prevent SQL injection. Also, all the data displayed for each event is retrieved from the “events” table in the database.

In my database, I have a table called “events” that stores all the required information for each event. The table has a primary key named “event_id”, which is of the data type integer and auto increments for each event. The columns “category”, “name”, “place” and “organiser" are all of the data type varchar with a maximum length of 256 characters and are set to “Not Null”, as this is the information that every event must-have. The column “date” stores the date of the events, which is used later to order the events by date on the home page. The “description” column is also of the data type varchar and the maximum length is 8000 characters. The “extra_info” column is also of the data type varchar with a maximum length of 5000 characters. Like the “date” column, the “description” and the “extra_info” columns are also set to “Not Null”.

The “userinfo” table is used to store the details of the students when they have registered. It has three columns “email_id”, “password” and “name”, with “email_id” set as the primary key for the table. All three columns are set as varchar with a maximum length of 256 characters and have the “Not Null” constraint as all 3 columns are required for the user to be registered.

The “bookings” table stores all events the students have booked. It has the primary key “booking_id” which is of the data type integer, it is set to “Not Null” and it is auto incremental so each booking will have its own unique id so it can be easily accessed. When each student books the event, the booking table also stores their email address and the event ID. Therefore, the bookings table has two foreign keys called “email_id” and “event_id”, which create a relationship with the “userinfo” and the “events” table.

Similarly to the “bookings” table, the “likes” table also has a primary key called “like_id”, which is also auto incremental and not null. This table also stores the students' email ID and the event ID, which are foreign keys connecting the table to the “userinfo” and the “events” table, respectively.

As a result of these foreign keys, the “userinfo” table and the “events” table have a many-to-many relationship, as a student can book and like many events, and events can have multiple students.

The database adheres to the 3rd normal form, as it does not have partial and transitive dependency.

All the tables in the database have already been populated, and I have already registered 4 users: Alice, Brad, John and Selena which you can find in the “userinfo” table. Their e-mail address consists of their name with “@aston.ac.uk” at the end, and their passwords consist of their name with “123” added on. For example, Alice’s e-mail address is “alice@aston.ac.uk”, and the user’s password is set to “alice123”. All the users have already booked/liked at least one of the events as you may find their data in the “bookings” and the “likes” tables.

In conclusion, my finished project meets all the required functionalities. The basic information about each event can be seen as you scroll down on the home page. My project meets the 2nd and the 3rd requirements as the user can choose to see each event by category or by their date using the drop-down menu on the home page. The user can also click on the “More Details” button to view additional information about the event and show their interest in the event, hence it also meets the 4th requirement. Finally, the user can also click the “Book Now” button on the events page to successfully book onto any event which completes the required functionality.

Reference List:

Brian Design (2020) HTML CSS JavaScript Website Tutorial– Responsive Beginner JS Project with smooth scroll. [online video] Available at: https://www.youtube.com/watch?v=3-2Pj5hxwrw&list=PLvxuqxA4YloC3UlYuqL1fx9iGjJmkWlmk&index=3&ab_channel=BrianDesign .