Name: Mukitur Rahman
Student ID: 200080734
Website url: http://200080734.cs2410-web01pvm.aston.ac.uk/aston%20web/index.php

Website report

Website structure

Before getting more in depth into the website that I have designed, I will be briefly describing how I have structured it. So, the whole project folder (aston web) was divided into sections in order to keep everything neat and organised, so that I could update and maintain it easily throughout the development of the project. Most of the CSS code was separated from the HTML code of the pages and put into a separate sub-directory, inside "aston web", called "css", similarly I have done the same with the Images (put into "images" sub-directory), fonts files (put into "fonts" sub-directory), JavaScript(put into "javascript") and all the website php code for functionality and communication with database was also put into a "php" sub-directory.

HTML5 was used to design the website and CSS was used to control the layout e.g., positioning the information on the screen, furthermore, I have made use of JavaScript code in order to show error messages to students, specifically JavaScript was mainly used for the data validation of the forms contained in the registration, login and booking pages.

The following is a summary of the various pages of the website:

- **index.php**

I have kept the navigation system of the website neat and simple, a student must login into the website before accessing its main contents, therefore I have set the login page as the actual index page of the website. Furthermore, if they are not already registered, I have provided a link to the registration page inside the index page. The actual page is very simple it features a login form, which accepts an aston email address and password. After successful login they would be automatically redirected to the actual homepage.

- **registration.php**

Students would be able to reach this page using the hyperlink contained in the index page. This would feature a registration form where students must pass all the validation in order to be later recognized for the login process. Furthermore, on successful registration it will redirect users to the login page.

- **homepage.php**

This would be the page where users will be redirected to after successful login, this would feature a navigation bar containing links to the events page and a logout. This page doesn't feature any other functionalities other than providing navigation, however it acts as a welcome page to the users successfully logged in.

- **events.php**

once in the events page users would be able to select from a total of nine events links (three for each category), which are graphically separated into the three sections as required from the assignment brief which are: Sport, Culture and Other, additionally underneath each event there would be a like button for students to show interest once clicked it would redirect them to the booking page as if they are interested they might want to join the event too. Furthermore, once a

link is clicked it would redirect the user to a event details page which would contain lots of information regarding that specific event.

- **logout.php**

this page's main purpose is to show users that they have been successfully logged out from the website.

- **eventdetails.php**

this page would hold all the information for each event such as event description, event organiser and his details, start time and end etc., furthermore, the page would feature a flexible background, meaning that it would contain different background images according to the select event. Also, I have included a "book here" button which would redirect users to the booking page, if they are interested to join that particular event.

- **Booking.php**

This page can be reached either by clicking the "like" button from events page or from the "book here" button from event details page, this would feature a simple booking form where students must input their Aston email address in order to successfully book into the previously selected event.

Those were all the pages of my actual website; I have kept those files separate from the actual functionality sections of the project, which are implemented by the following documents:

- **function.php**

this is the main file that contains most of the functions coding for the website, it features a "createUser" function (for the registration feature), which simply inserts data into the "students" table of the database, also, it hashes the password values, before putting them into the database. For protection against XSS attacks or data breaches.

Furthermore, for the login system, it features a "userNameExists" function which checks from the "students" table records if a username is registered and then a "successfulLogIn" function which uses it to authenticate a student through email and password values.

Additionally, I have used a "gatherData" function, in order to collect information regarding a specific event (depending on the URL "Id" value) from the "eventdetails" table in the database. This function would then be called by the eventdetails.php in order to portray the gathered information on the actual event details page.

Furthermore, for the booking system I have included a function called "bookingUser" which would simply insert student data and event details (those data are passed to the function by "booking.php") into the "reservations" table.

Additionally, in order to implement the like system, I have included two functions, which are "showLink" which would simply gather "eventId" and "eventType" details from the database table called "eventdetails" and put them into the URL, then a "likeGet" function which would, insert that "eventId"

and “userID” (gotten from the session) values into the “likeCount” table of the database, this way we would know which users like which event.

Furthermore, I have included server-side validation in addition to JavaScript (talked about later), in order to better protect, from unauthorised users and hackers. Specifically, the following are the function I have implemented for the validation.

1. invalidEmail

   checks if email is a correct aston email address.

2. invalidValue

   checks that phone number contains only numbers.

3. ppattern

   checks that password is strong. It must contain uppercase, lowercase letters a number and at least 8 characters.

4. passVal

   checks that password is equal to the confirmation password.

5. lengthPhone

   checks that phone number is valid by checking the length (10 chars allowed)

- **booking.php**

this document simply gets data from the booking page and event details page and passes them into the “bookingUser” function in “function.php”, in order to register users booking to a specific event.

- **connection.php**

this is the document that makes the connection to the database possible, it would pass database login information to “mysqli_connect” function which will connect to the Database. This is stored into a $conn variable, which is later used for every connection to the database, such as to fetch data from the “eventdetails” table.

- **flexback.php**

this document implements the flexible background image in the “eventdetails.php” page, this would use the “id” value of the link in event page in order to distinguish each event link, and from this it would decide which background to set, additionally, it would cycle through all the events through if statement.

- **lg.php**

this file simply passes the login values taken from user input to the “successfulLogIn” function, so that if the values are found in the “students” table of the database, they would be successfully logged in, and be redirected to the actual welcome page (“homepage.php”).

- **like.php**

this simply calls the “likeGet” function in “function.php” and passes the “eventId” and “userID” values to it, in order to register the like correctly into the “likeCount” table in the database.

Name: Mukitur Rahman
Student ID: 200080734
Website url: http://200080734.cs2410-web01pvm.aston.ac.uk/aston%20web/index.php

- **reg.php**

it simply holds the data from the registration form, puts them into local variables and passes them to the "createUser" function in "function.php", this would allow the registration of a student. It would also pass the form values to the validation functions that I have described earlier in "function.php"

- **booking.js**

this script is used by the "Booking.php" page, this would check if the email input by the users is a valid Aston email address also, it will check that the input box is not left blank, if any of those test fails, the booking request to the server would not go through. This way we can reduce requests send to the DB, therefore reduce slowdowns.

- **login.js**

this script is used by the "index.php" page, this would simply check that either email or password field are left blank.

- **registration.js**

this script is used by the "registration.php" page, and it performs a number of input validation, such as: fields not being left blank, name or surname must not contain numbers or special characters, valid Aston email, no special characters or letters allowed on student id and phone number, valid phone number pattern, confirming password must match with the password and it must contain a number, uppercase and lowercase letter.

If all those validations are passed, then the user data would be registered in the "students" table of the database otherwise error messages would appear on the bottom area of the form, describing clearly the error.

Furthermore, as I have said earlier along with functions and scripts, I have also separated CSS files from the actual HTML, the following are the CSS files:

- booking.css
- eventdetails.css
- events.css
- homepage.css
- index.css
- login.css
- logout.css
- myfonts.css
- registration.css

as you can see, I have put an index.css which would be used by all of the pages on the website, which would define a consistent background structure for the pages. This was done also, so, that I did not have to repeat code unnecessarily, for e.g., it contains CSS for the navigation bar and footer.

database structure and functionality

for the Database part, I have created four tables using SQL language, they are the following:

Name: Mukitur Rahman
Student ID: 200080734
Website url: http://200080734.cs2410-web01pvm.aston.ac.uk/aston%20web/index.php

- students

this table would be used for the registration process, it would contain all the registration data of the students which would be inserted through the “createUser” function. Also, it would have feature a “userID” value as primary key which would be used as foreign key for the “reservations” and “likeCount” tables. One student can place many bookings (one to many relations with reservations table), one student can have many likes (one-to-many relation with likeCount).

- eventdetails

this table would be used to store details for each event, it would be then used by the “gatherData” function, in order to fetch specific event data. Also, the table features a primary key called “eventId”, which would be used as a foreign key in the “reservations” and “likeCount” tables. It has a one-to-many relation with the “likeCount” because one event can be liked many times, and a one-to-many relations with “reservations” as one event can be reserved many times.

- likeCount

this table would have a primary key called “likeID” and would feature two foreign keys which are: “eventId” and “userID”, also, it would be used mainly to register user’s likes to specific events.

- reservations

This table would feature a primary key, “bookId” and two foreign keys which are: “userID” and “eventId”, we included those so that we would know which user liked which event. Furthermore, it is used for the implementation of the booking system, data are inserted into the table via the “bookingUser” function in function.php.

Furthermore, I have constrained all the tables above with correct data types, for e.g., date and time values were set to date and time type, therefore, we would not find wrong data formats inside those columns. I have used Int type for the primary keys and VARCHAR for all the other data columns.

constraints/fallbacks of the website/additional features

one of the major fallbacks, is the issue with the like system, as one user can keep liking an event without a proper restriction, an idea I had was to change the button into a “dislike” button once a user already liked an event, so that button could delete the like row in the “likeCount” table once clicked, however because of time constraints I wasn’t able to implement this feature.

A major advantage of the website is the resizability, for e.g. when zooming out or in all parts resize appropriately. Thanks to appropriate controls from CSS.