**Theatre Booking System**

**Assessment Objectives**

Create a theatre booking system using PHP, JavaScript, HTML5, CSS, and MySQL.

**Requirements**

Your Web Based theatre booking system should conform to the following requirements:

* **Task 1 : Present a list of productions, loaded from the current state of the database.** You should create two files (mystyles.css and index.php. In mystyles.css you should include all your styling rules. In index.php you should generate a html page with some headings, information about the theatre, and a list of productions. The list of production must be dynamically retrieved from the database. For each production, display a button (which will be used/extended in the next task). If you wish you can add a few images that should be in a folder called "images". See below a sample snapshot of how the (minimal version of your) first page may look like.

**Hints:**

* + This task involves use of CSS, HTML, PHP, and MySQL.
  + You can use the following MySQL query, which you need to adapt to and appropriately handle in your php script

SELECT \* FROM Production;

* **Task 2: Allow the user to select a production and see the performances for that production.** You need to modify index.php so that it allows a user to select a production and invoke another resoruce perf.php to which you need to pass all necessary data (i.e., title and basic ticket price). File perf.php should display some headings, and a list of all performances for the selected production. See below a sample snapshot.

**Hints:**

* + This task involves use of HTML, PHP, and MySQL (possibly also CSS -- if you need to extend or modify the styles of your web application).
  + A useful query for this part, to adapt to your PHP script, is this one

SELECT \* FROM Performance WHERE Performance.Title = 'Cats';

In your solution, the name of the production should be the one selected by the user in the previous page (produced by script index.php).

* **Task 3 : Allow the user to select a performance and the available seats and their prices from the current state of the database.** You need to modify perf.php so that it allows a user to select a performance and invoke another resoruce seats.php to which you need to pass all necessary data (i.e., title, basic ticket price, date and time of the selected performance). File seats.php should display some headings, and a list of all seats available (only the available ones!) for the selected performance.

**Hints:**

* + This task involves use of HTML, PHP, and MySQL (possibly also CSS -- if you need to extend or modify the styles of your web application).
  + A useful query for this part, to adapt to your PHP script, is this one

SELECT Seat.RowNumber, Zone.PriceMultiplier \* 15.00   
FROM Seat, Zone   
WHERE Zone.Name=Seat.Zone   
AND Seat.RowNumber NOT IN   
(SELECT Booking.RowNumber FROM Booking   
WHERE Booking.PerfTime='16:00:00'   
AND Booking.PerfDate='2015-12-04');

In your solution, the data (seat, performance time, price multiplier, etc.) should be relative to the production and performance being selected by the user.

* **Task 4: Allow the user to select one or more seats.** You need to modify seats.php so that it allows a user to select one or more seats and:
  + click a "check" button to see, in an alert window, a summary of the selection and the total price (calculated client-side)
  + enter an email address and press a "book" button to invoke another resource book.php.

See below a sample snapshot for tasks 3 and 4.

**Hints:** This task involves JavaScript (as well as PHP, HTML and possibly also CSS). At this point, you should have all data you need to show a summary just with client-side code, without performing additional queries to the database.

* **Task 5: Insert the booking in the database and display a confirmation.**

This task involves use of HTML, PHP, and MySQL (possibly also CSS). You need to edit book.php so that it updates the database with the new bookings, and display a confirmation of the purchase. At this point a real system would invoke a payment system. You may ignore this and end the process just by displaying a confirmation message.

**Hints:**

* + This task involves use of HTML, PHP, and MySQL (possibly also CSS -- if you need to extend or modify the styles of your web application).
  + A useful query for this part, to adapt to your PHP script, is this one

INSERT INTO Booking VALUES ('ZP@email.com','2017-11-01','19:00:00','Z18');

In your solution, the booking values should be relative to the actual booking being made.

**Summary of Files**

Your solution should consist of the following files/folders:

* mystyles.css : with all your styling rules
* index.php : with the html and php scripts to address requirement 1.
* perf.php : with the html and php scripts to address requirement 2.
* seats.php : with the html, javascript, and php scripts to address requirement 3. except the update of the database
* book.php : to insert the booking in the database and show a confirmation
* connect.php (optional) : with the function to connect to the database, which you can reuse in all the pages.
* images (optional folder) : with images.

**The database**

We must use, as a database, the solution to Assessment 1. We recall the relational schema below.

Production (Title,BasicTicketPrice)

Performance (PerfDate,PerfTime,*Title*)

Zone (Name,PriceMultiplier)

Seat (RowNumber,*Zone*)

Booking (Email,*PerfDate*,*PerfTime*,*RowNumber*)

Feel free to add productions, performances, and bookings. Make sure that there is a fair number of seats we can book, when testing your solution!

[Top](https://moodle.kent.ac.uk/2018/pluginfile.php/106572/mod_resource/content/4/Assessment2.html?embed=1)

**More Tips**

A good solution will combine what you have learnt in HTML, CSS, JavaScript, PHP, and SQL.

We are looking for clear, concise, and commented code.

Make use of javascript validation, server side (php) validation to check against input.

Ensure your database is not vulnerable to SQL injection attacks. To do so, make use of the functions available to PHP and MySQL.

Do not make unnecessary queries to the database. Recall we have seen several ways to transfer information across the pages (superglobals $\_GET and $\_POST, hidden fields, and sessions). Make the best of them!

We want the website to look nice, so use CSS, however, we are not expecting amazing CSS design, just enough to make it sensible, and usable.