A list of steps to be used for the stage 1 diagrams by Vern Watkins

(also used to start planning how the Requirement Description Document and the Software Requirement Specification for Stage 1 will be made)

=== Step 1: Make a beginning theatre webpage with connectivity to a table in a MySQL database and do some basic testing as follows:

substep 1-1: Make a webpage with 4 buttons (to represent 4 seats, labeled 1A, 1B, 1C, and 1D) and 'publish' the webpage via a website hosting provider. Insert a textbox at the bottom and code the buttons so that clicking them will print their corresponding seat number in the textbox: "You have requested seat 1B", or "You have requested seat 1B", etc. (with the textbox being tall enough so that all messages will show).

substep 1-2: From our account with our hosting provider, create our database that our hosting provider supplies.

substep 1-3: Include code in our webpage to connect to the database whenever our webpage is opened.

substep 1-4: From our account with our hosting provider, make a table that can be used to do some basic testing. The [customers] table looks like a good one to make first (since we have to make it sometime anyway).
For a [customers] table, the columns are:

```
CREATE TABLE customers (
customer_id int PRIMARY KEY,
first_name varchar(30),
last_name varchar(30),
address varchar(50),
city varchar(50),
state varchar(50),
zip_code varchar(10),
telephone varchar(12),
email  varchar(50),
age int(3)
);
```

From our account with our hosting provider, enter some test data for the first customer.

```
INSERT INTO customers (first_name, last_name, address, city, state,
zip_code, telephone, email, age);
VALUES ('John', 'Doe', '123 anystreet', 'Amarillo', Texas, '79106', '806-
555-1212', '50');
```

substep 1-5: To test that we can make our database report something, use a sql query command in our webpage, executed by a new button1, to show the information of the first test customer in the output textbox.

```
SELECT *
FROM customers
WHERE = customer_id = '1';
```

substep 1-6: To test that we can make our database enter new data, add another new button2 to the webpage so that when it is clicked, it will make a new row in the customer table (using a "insert into" statement) and will put whatever is typed in a new textbox (textbox2) (to use as input) into the first_name column.

```
INSERT INTO customers (first_name)
VALUES ('Susan');
```

substep 1-7: Then to see this new data, add to the webpage another button (button3) that uses a different sql query command to show the first name of the added customer in the output textbox.

```
SELECT first_name
FROM customers
WHERE = customer_id = '2');
```

===Step 2: Make more of the required tables, and modify the theatre on the webpage to be more like the completed theatre.

substep 2-1: Make the [statusofseatsplay1] table (more plays will be handled in later project steps).
This table will have 96 rows and 6 columns. The columns are:
- IDkey (1,2,3,4,5...96, which also associates the seats with integers),
- for each 'row' of seats (1-8; row 1 are the seats closest to the stage, row 8 are those farthest from the stage) (bit),)
- for each 'column' of seats (A,B,C,D...L; column A is to the right of the stage, column L is to the left of the stage) 'varchar' (which are any numbers, letters and special characters)
- the alphanumeric designator for each seat (1A, 3B or 12L, etc)(varchar),
- the status of the seat: 0 is empty and 1 is reserved (bit)
- the price of each seat (assigned by the administrator, make all seat prices '30' for now) (integer)

```
CREATE TABLE statusofseatsplay1 (
   seat_id int(2)        PRIMARY KEY,
   row bit(1)            NOT NULL,
   column varchar(1)     NOT NULL,
   designator varchar(2) NOT NULL,
   status bit(1)         DEFAULT 0,
   price int(3)          DEFAULT 30
);
```

An example of Row and Column values in the [statusofseatsplay1] table (where seats 1B, 2A and 12L have been selected):

```
 1: 1 A 1A 0 30
 2: 1 B 1B 1 30
 3: 1 C 1C 0 30
.. .. ..
 9: 2 A 2A 1 30
10: 2 B 2B 0 30
.. .. ..
.. .. ..
.. .. ..
```

89: 12 A 12A 0 30
90: 12 L 12L 1 30

(a guide to identify rows with seat designations in the [Statusofseatsplay1] table, just to show which rows represent which seats)

```
A  B   C   D   E   F   G   H   I   J   K   L
  +8  +8  +8  +8  +8  +8  +8  +8  +8  +8  +8
1  9  17  25  33  41  49  57  65  73  81  89
   ^                                       ^
   starts 2A                               starts 12A
```

(note, in this and the following table, row 1 represents seat 1A, row 8 represents seat 1L, row 9 represents seat 2A, row 10 represents seat 2L, row 89 represents seat 12A, and row 96 represents seat 12L, the last seat)

substep 2-2: Make a [seatspercustomer{insert customer number here}play1] table for each customer that requests seats (done right after registration). The customer number is obtained from the customers table. This table will have 96 rows and 6 columns. The columns are:
- IDkey (1,2,3,4,5...96, which also associates the seats with integers),
- representations for each 'row' of seats (1-8; row 1 are the seats closest to the stage, row 8 are those farthest from the stage) (bit),
- representations for each 'column' of seats (A,B,C,D...L; column A is to the right of the stage, column L is to the left of the stage (varchar),
- the alphanumeric designator for each seat (1A, 3B or 12L, etc)(varchar),
- the customer ID that is requesting seat purchases,
- the pending status of the seat: 0 is not chosen and 1 is desired (bit)
- the verified status of the seat: 0 is not chosen and 1 is paid for (bit)

```
CREATE TABLE seatspercustomer{insert customer number here}play1 (
   Seat_id int(2)          PRIMARY KEY,
   Row bit(1)              NOT NULL,
   column varchar(1)       NOT NULL
   designator varchar(2)   NOT NULL,
   customer_id varchar(4) DEFAULT XX,
   pending bit(1)          DEFAULT 0,
   verified but(1)         DEFAULT 0,
);
```

substep 2-3: On the webpage, create a grid of the seats, representing seats for 8 rows (1-8) and 12 columns (A-L), to query the status of each of the 96 seats to the respective row in the [statusofseatsplay1] table.

Add code to the webpage so that if a customer selects a seat that is available in the [statusofseatsplay1] table (value being a 0), a "You have requested seat {seat designation}" will show in the textbox, and if not available (value being a 1), to show a different message: "Seat {seat designation} is not available", and the customer can continue choosing seats, or go to the checkout page. Add a Checkout button to the webpage to the right of the textbox.

Apply color to the buttons such that green indicates that seat is available, and red indicates that seat has been reserved and paid for (by any customer). A nice touch for the output textbox is to make its inner color to be a slight shade lighter than the general page background color, instead of "white". Also, always add a line after the last 'seat' message saying: “To submit your requests, please proceed to the Checkout.”

substep 2-4: From our account with our hosting provider, in the [statusofseatsplay1] table, modify the status of seat 3C to be reserved (value of 1), and check that on the webpage when the user clicks on seats 1A, 3C, and 12H, that these 3 lines will show in the message box:
"You have requested seat 1A"
"Seat 3C is not available"
"You have requested seat 12H"

substep 2-5: When a seat is requested on the webpage, use a sql update command to change the seat’s column 5 ‘pending’ value in the [seatspercustomer{cust#}play1] table from 0 to 1 (has now been requested), and from our account with our hosting provider, look at the table to check that this value in the [seatspercustomer{cust#}play1] table has been changed.

substep 2-6: Make a “Checkout” webpage that the Checkout button opens, titled “Checkout”, with showing the customer name (referenced from the [customers] table), seats requested (ex, 1A, 3C, and 12H, referenced from the [seatspercustomer{cust#}play]1 table), and the total price of the *pending* seats referenced from their prices in the [statusofseatsplay1] table. When the customer pays, each of their pending seats will again be checked with the status in the [statusofseatsplay1] table, and if *still available, 1. the status of that seat in the [statusofseatsplay1] table will be ‘reserved’, and 2. the value in the 'verified' column of the [seatspercustomer{cust#}play1] table will be set to 1.

If when paying, for any pending seats suddenly becoming not available in the [statusofseatsplay1] table (due to someone else buying them while the customer is still selecting seats), the value in the column 6 ‘verified’ column will be set to 0 and a message will show (just like in substep 2-3) “Sorry, seat {seat designation} has been reserved”, and the total price will be recalculated from the remaining verified seats in the [seatspercustomer{cust#}play1] table, and “Your readjusted total price is now $____.”

Items for a future Step 3:
make a registration webpage (which will generate a [seatspercustomer{insert customer number here}play1] table.
..and make login/password pages for the customer and the administrator, with their respective login/password tables.
..and make a [allplayswithprices] table
..and add the ability for a customer to click on a previously selected seat to remove it from their list of seats.