V.Watkins Gantt chart preparation of some beginning steps

(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 4-seat theatre with connectivity to a MySQL database of several tables as follows:

substep 1-1: Make a webpage with a 2x2 grid of 4 option buttons to represent seats A1, A2, B1, and B2. Insert a textbox at the bottom and make its color (inside it) to be a slight shade lighter than the general page background color, instead of "white"

substep 1-2: Using the hosting providers supplied MySQL, make a Database containing a [statusofseatsplay1] table for play 1 (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 represents each seat)
- column representations for each 'COLUMN' of seats (A,B,C,D...L; A is at the left side, L is at the right side) 'varchar' (which are any numbers, letters and special characters)
- row positions (2) (bit) (a 'BIT' is integers 1-64) (1 is the closest to the stage, 8 is farthest away)
- the alphanumeric designator for each seat (A1, B3 or L12, etc) (varchar)
- the status of the seat: 0 is empty and 1 is reserved (by any customer) (bit)
- the price of each seat (assigned by the administrator, make all seat prices '30' for now) (integer)

```
CREATE TABLE STATUSOFSEATSPLAY1 (
   PRIMARY KEY (ID)       NOT NULL,
   COLUMN VARCHAR (1)     NOT NULL,
   ROW BIT (1)             NOT NULL,
   DESIGNATOR VARCHAR (2) NOT NULL,
   STATUS BIT             DEFAULT 0,
   PRICE INT (3)          DEFAULT 30
);
```

An example of Row and Column values in the [statusofseatsplay1] table (where seats A2, B1 and L12 have been selected):

```
 1: A 1 A1 0 30
 2: A 2 A2 1 30
 3: A 3 A3 0 30
.. .. ..
 9: B 1 B1 1 30
10: B 2 B2 0 30
.. .. ..
.. .. ..
.. .. ..
89: L  1 L1  0 30
90: L 12 L12 1 30
```

(a guide to identify rows with seat designations in the [Statusofseatsplay1] table, just to illustrate 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

(thus, row 1 represents seat A1, row 9 represents seat B1, row 18 represents seat C2, row 35 represents seat E3, and row 96 represents seat L8 <the last seat>)

substep 1-3: Check that clicking the upper left option button will print in the textbox: "You have requested seat A1", then clicking the lower right option button will add "You have requested seat B2" in a new line underneath (with the textbox being tall enough so that both messages will show).

substep 1-4: Make a [seatspercustomerplay1] table.
This table will have 96 rows and 6 columns. The columns are:
- IDkey (1,2,3,4,5...96 to represent all the seats),
- column representations for each 'level' of seats (A,B,C,D...H; A is the closest to the stage) (string values),
- the alphanumeric designator for each seat (A1, B3 or H12, etc) (string values),
- the customer ID,
- 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 SEATSPERCUSTOMERPLAY1 (
   SEAT_ID INT             PRIMARY KEY,
   LEVEL VARCHAR (1)       NOT NULL,
   DESIGNATOR VARCHAR (2)  NOT NULL,
   CUSTOMER_ID VARCHAR (4) DEFAULT XX,
   PENDING BIT             DEFAULT 0,
   VERIFIED BIT            DEFAULT 0,
);
```

substep 1-5: Using code, associate each of the 4 option buttons with their respective row in the [seatspercustomerplay1] table (rows 1,2,9,10 in this substep) such that when a seat is requested, change their column 5 'pending' values in the [seatspercustomerplay1] table from 0 to a 1 (has now been requested), and look at the table to check that these values in the [seatspercustomerplay1] table have been changed.

substep 1-6: Make a [customers] table. The columns are:

```
CREATE TABLE CUSTOMERS (
CUSTOMER_ID INT          PRIMARY KEY,
FIRST_NAME VARCHAR (30) NOT NULL,
LAST_NAME VARCHAR (30)  NOT NULL,
ADDRESS VARCHAR (50)    NOT NULL,
```

```
CITY VARCHAR (50)       NOT NULL,
STATE VARCHAR (50)     NOT NULL,
ZIP_CODE VARCHAR (10)  NOT NULL,
TELEPHONE VARCHAR (12) NOT NULL,
EMAIL VARCHAR (50)     NOT NULL,
AGE INT (3)            NOT NULL
);
```

===Step 2: Modify the Theatre on the webpage to be more like the completed theatre, and include more of the required tables.

substep 2-1: On the webpage, make the option buttons to look like squares, colored such that green indicates that seat is available, and red indicates that seat has been reserved and paid for (by any customer), according to their status in the [statusofseatsplay1] table. Next, create a grid of the new squares, representing seats for 8 rows (1-8) and 12 levels ('columns') (A-L), and associate each of them to their row in the [statusofseatsplay1] table just like was done in 'substep 1-5', and check that when the user clicks on a multiple of seats A1, B3, and H12, that these 3 lines will show in the message box:
"You have requested seat A1"
"You have requested seat B3"
"You have requested seat H12"

-And always add a last line saying: "To submit your requests, please proceed to the Checkout"

Also, add a 'Checkout' button to the webpage (I suppose to the right of the textbox).

Whenever a customer selects a seat, it will be checked with the [statusofseatsplay1] table, and if available, that seat will be assigned to column 5 of the [seatspercustomerplay1] table ('pending' set to 1). If not available in the [statusofseatsplay1] table, a message will be added to the message box saying: "Seat {seat ID} is not available", and the customer can continue choosing seats, or go to the checkout page.

substep 2-2: Make a "Checkout" webpage that the Checkout button opens, titled "Checkout", and showing the customer name (referenced from the [customers] table), seats requested (ex, A1, B3, H8, referenced from the [seatspercustomerplay]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 [statusofseatsplay1] table, and if still available, 1. the status of that seat in the [statusofseatsplay1] table will be 'reserved', and 2. the cell in row 6 of the [seatspercustomerplay1] table (verified) will be set to 1.

If after paying, for any pending seats suddenly becoming not available in the [statusofseatsplay1] table (due to someone else buying them immediately before the customer does), the row 6 'verified' cell will be set to 0 and a message(s) will show "Sorry, seat(s) {seat designations} have been reserved", and the total prices will be recalculated from the remaining row

6 verified seats in the [seatspercustomerplay1] table, and “Your readjusted total price is now $____.”

For a future Step 3: make a [allplayswithprices] table

-----note:
* "The user can also click on a previously selected seat to remove it from their list of seats."
- add that ability to the relation between each of the 96 seats in the more-completed webpage and the [statusofseatsplay1] table.