a). Short description of the final project

For our final project, we implement a login page for two user roles:
audiences and employees. After they have successfully logged in, both user roles can log out, which will redirect back to the login page.

As an audience, they need to enter the correct email and password to log in. If the information does not match the data in our database, a specific error message will occur to help the user retry. Once they have successfully logged in, the website will redirect to their personal profile page. Additionally, the audience can view all the ticket information they have purchased and the corresponding event information. The total quantity of tickets is also visible to the audience.

As an employee, they need to enter the correct employeeID and password to log in. Similar to the audience, the error message will occur when an incorrect employeeID or password has been entered.

Firstly, employees can check the future schedule including the management time and the specific working location.
Secondly, employees can manage the event information including viewing current events, updating event information, inserting new events, and deleting the current events. The quantity of tickets sold for each event is also visible to employees.
Lastly, employees can view the player table, then search by the provided player information to see which event players participate in. In addition, employees can check if there are some players who attend all events.

b). how does your final Schema differ from the Schema you turned in

Comparing the two schemas, there are no significant changes in functionality. However, we added more attributes into our tables to help us accomplish the functionalities. For example, we add password and user type to help us implement the login function.

c). Why is the final Schema different?

We have two main reasons for changing our schema. Our first reason is we need to add attributes to our relation—-”purchase from ", in order to form a relational table. Without the relation attributes, our entity table and relational table will be the same. Thus, we add attributes to our relation entity to distinguish them. In order to divide the audience or employee character, we added the “type” attribute; in addition, we added the “password” attribute into three characters. The other reason is after learning normalization, we divided the big tables into a few smaller tables in order to make them more clear and independent.

As we work on functional dependency and normalization, we realize that the attributes we created in milestone 2 are not enough and some cannot form a functional dependency. Thus, based on the milestone 2 we decided to add more attributes to our

relational table in milestone3 in order to better construct the functional dependency and normalization.

d) SQL query requirements list

Note: some categories of the query are implemented more than once, however only one example is shown below

Insert Operation	insert into `Event_hold_in` (eventID , event_date , event_name , event_time , gym_number) values ('$id' , '$date' , '$name' , '$time' , '$gym')(eventInsert.php)
Delete Operation	delete from `Event_hold_in` where Event_hold_in.eventID = '$id'(eventDelete.php)
Update Operation	update `Event_hold_in` set event_name='$name', event_date='$date', event_time='$time', gym_number='$gym' where eventID = '$id'(eventUpdate.php)
Selection	SELECT * FROM Players, Perform_in WHERE Perform_in.playerID = '$search' AND Players.player_name = '$name'（search.php）
Projection	SELECT eventID FROM Event_hold_in; SELECT event_name FROM Event_hold_in; SELECT event_time FROM Event_hold_in; SELECT event_date FROM Event_hold_in; SELECT gym_number FROM Event_hold_in (projection.php)
Join Query	SELECT * FROM `players`,`perform_in` WHERE players.playerID = perform_in.playerID(playerNew.php)
Aggregation Query	SELECT COUNT(t.ticket_number) FROM audience2 a2, buy b, ticket_belong_to t, event_hold_in e where a2.userID = '$id' AND a2.user_receipt = b.user_receipt AND b.ticket_number = t.ticket_number AND t.eventID = e.eventID (myticket.php)
Nested aggregation with group-by	SELECT eventID,COUNT(t.ticket_number) FROM ticket_belong_to t GROUP BY eventID (countEvent.php)
Division query	SELECT playerID, player_name FROM `players` WHERE NOT EXISTS((SELECT eventID FROM `perform_in`) EXCEPT (SELECT eventID FROM `perform_in` WHERE players.playerID = perform_in.playerID)) (all.php)

Screenshots of our project:

Audience Login Page:

Login page(Image 1.1):

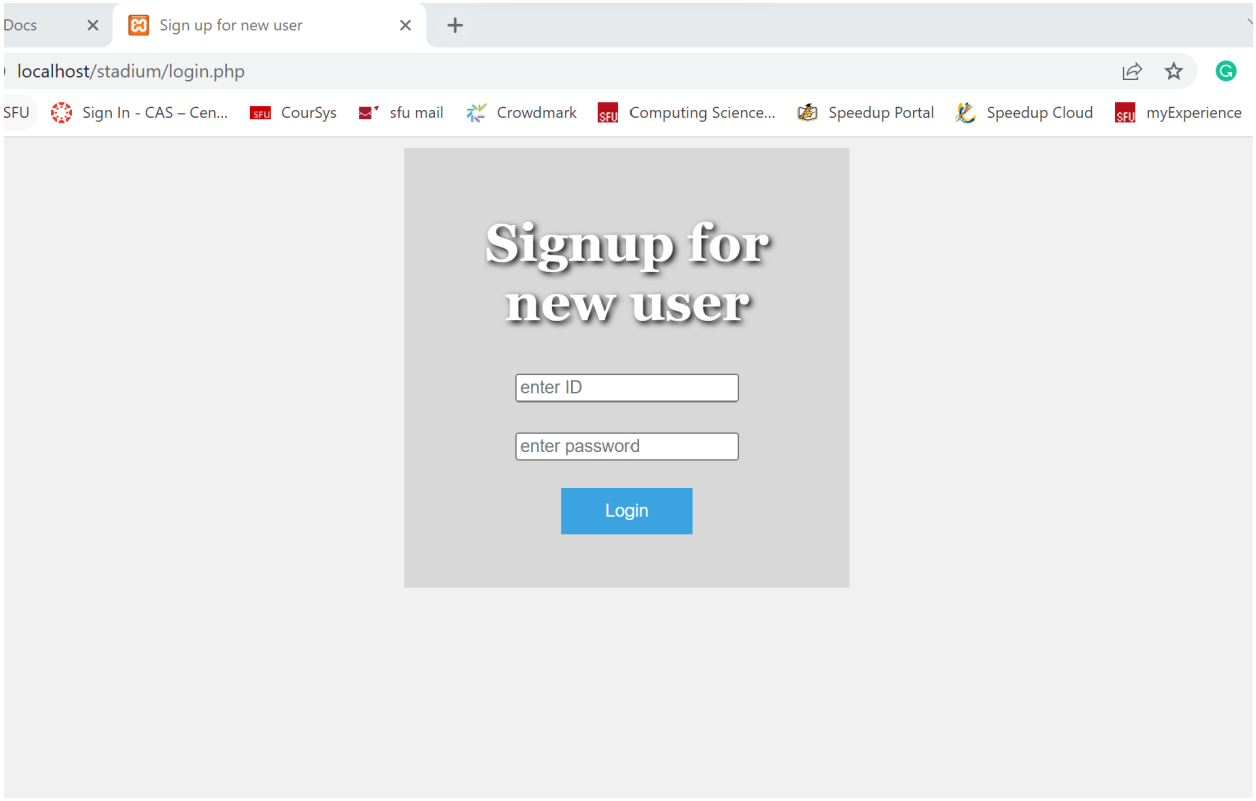

Image 1.1

Once the audience successfully logs in, they can click on the “Profile” button to view their personal information (given example illustrate when Emma is logged in, Image 1.2):

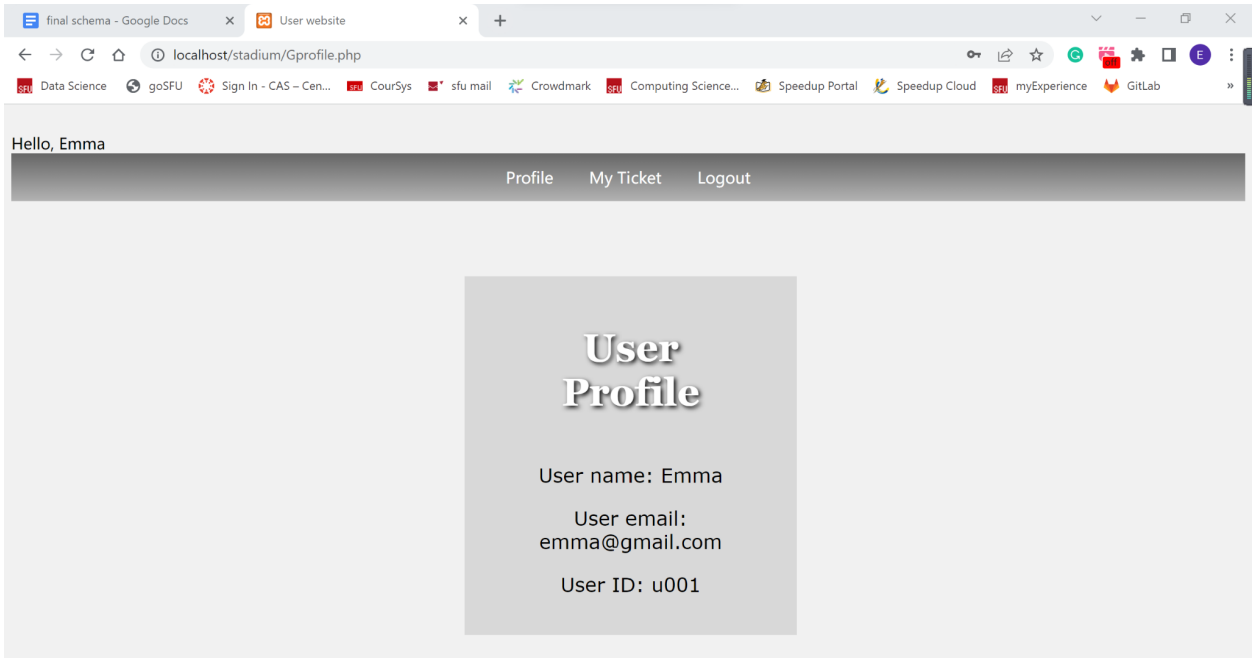

Image 1.2

Click on “My Ticket” to view all the current tickets (Image 1.3):

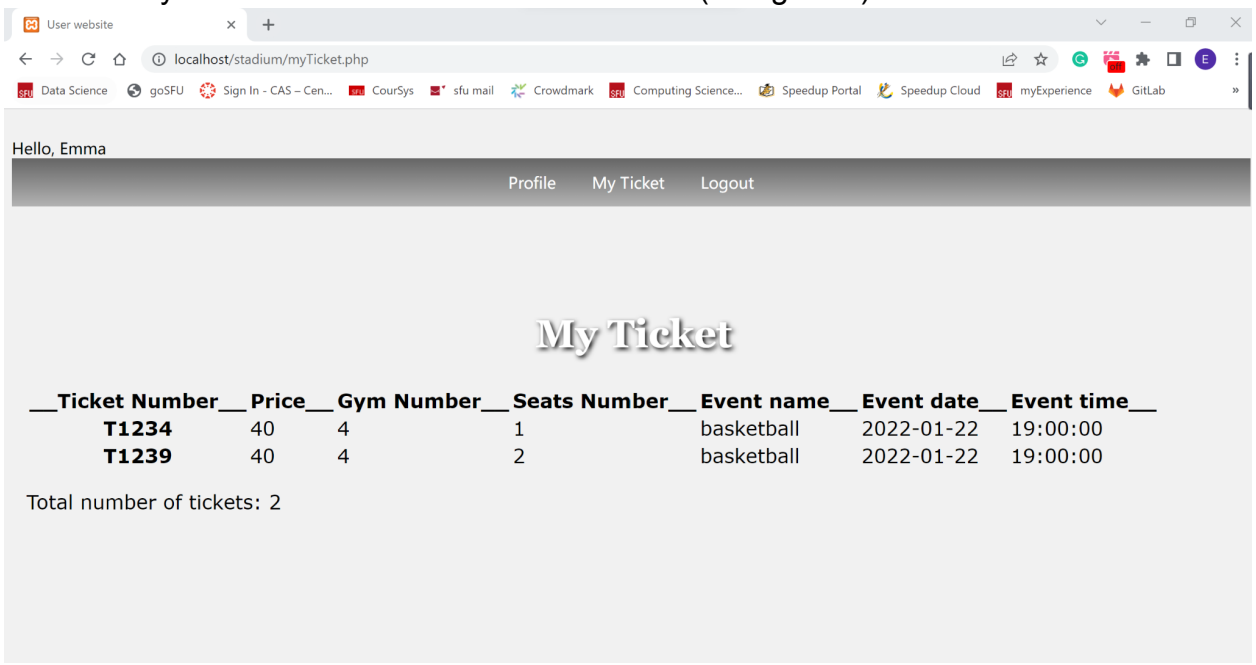

Image 1.3

Employee Login Page:

Login(Image 2.1):

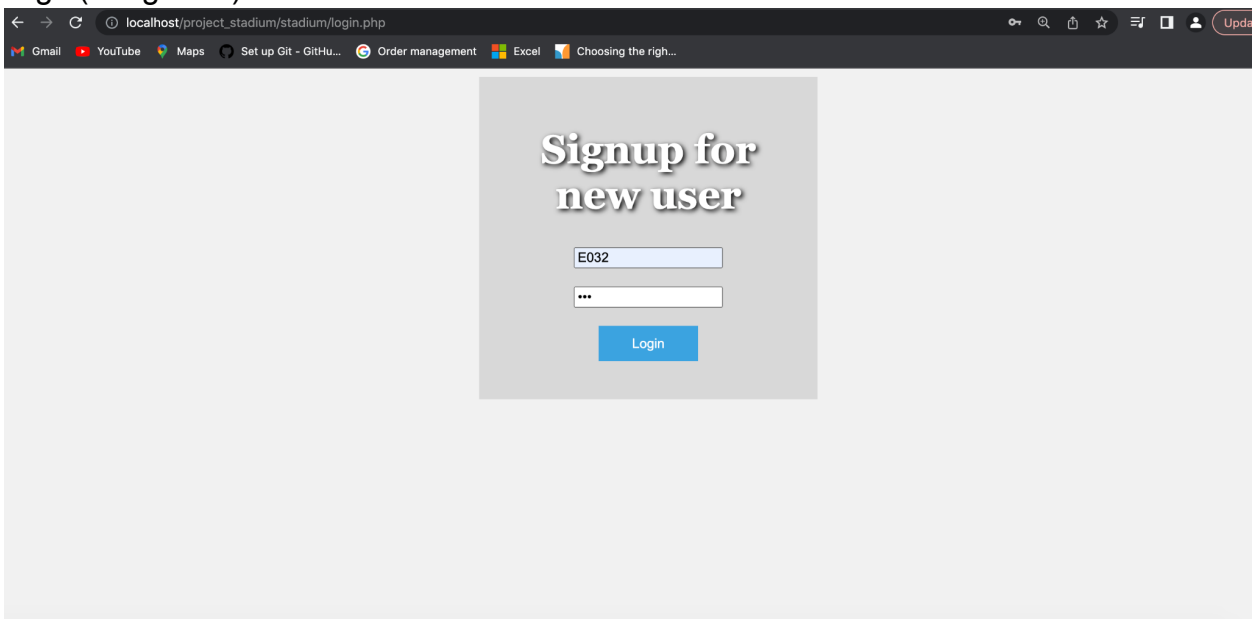

Image 2.1

Once the employee successfully logs in, they can click on the “Profile” button to view their personal profile (given example illustrate when Jelly is logged in,Image 2.2):

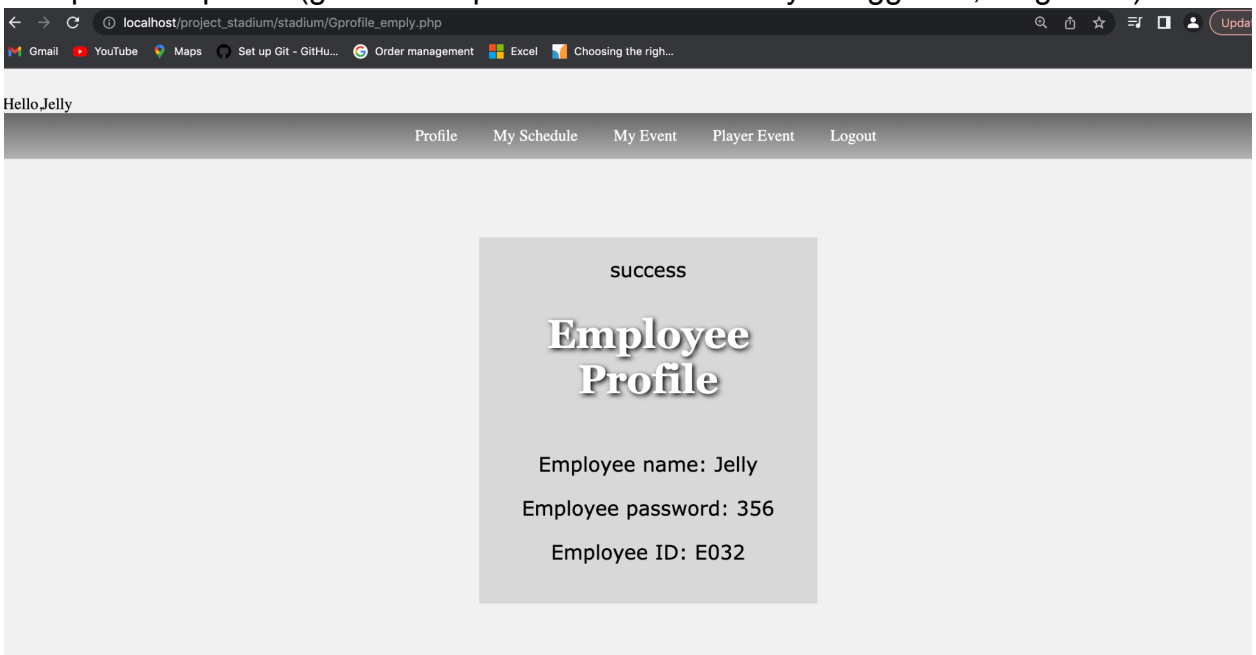

Image 2.2

The employee can also click on the “My Schedule” button to view their personal management information such as manage time, event ID, event name, job title and location of the event (Image 2.3):

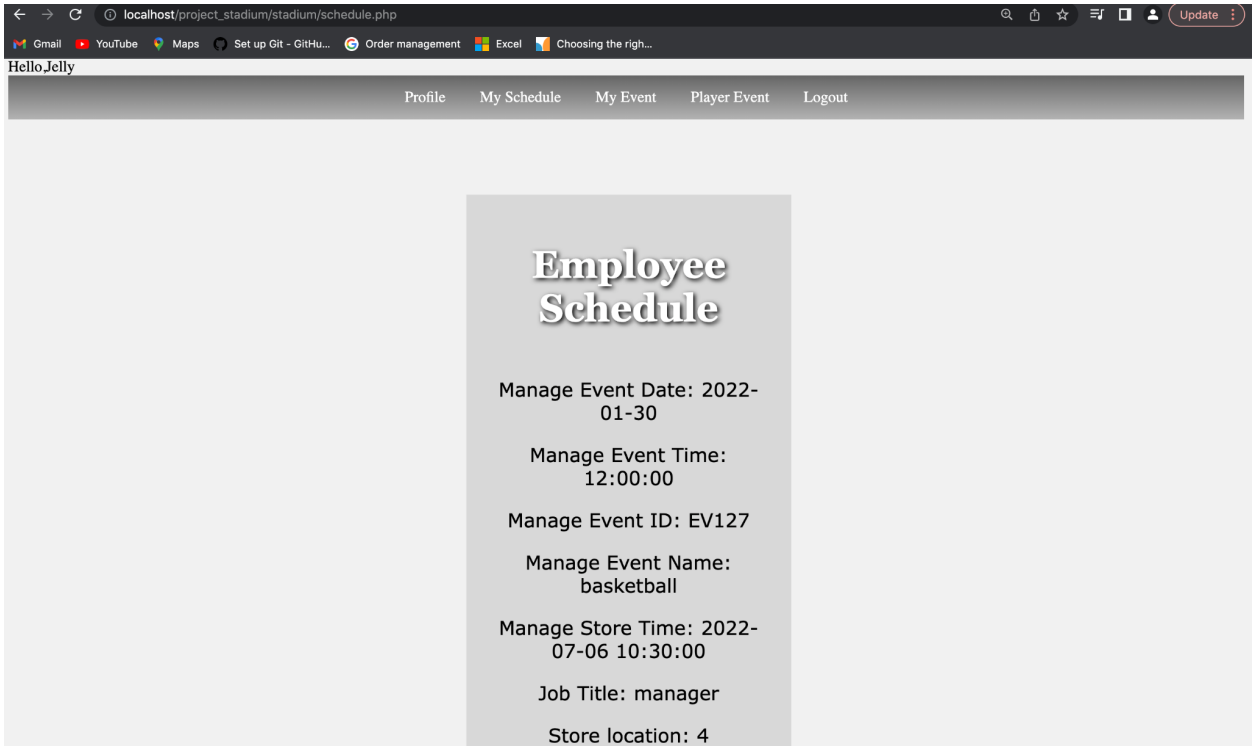

Image 2.3

Employee can also click on the “My Event” button to view all the future event information(Image 2.4):

- Employees can perform the following functionality: delete, update, insert, search.

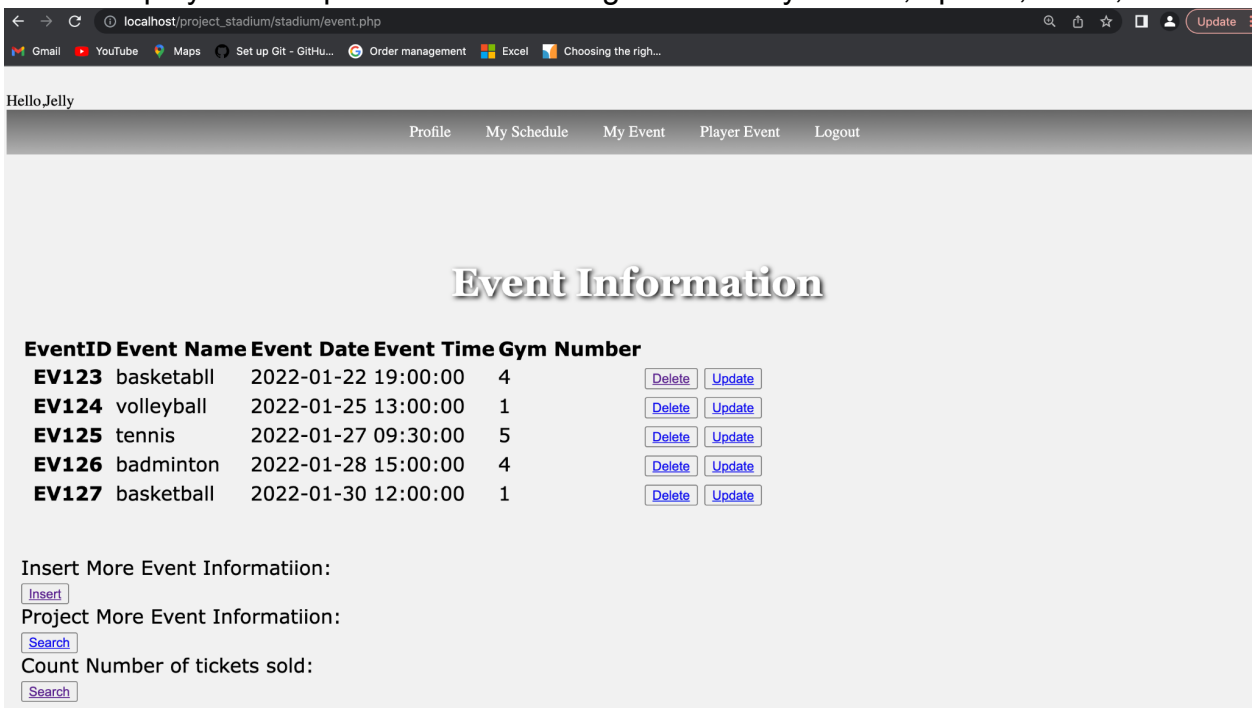

Image 2.4

Employee can click on the “Delete” button beside each event to delete selected row event(Image 2.5):

- For example, click on the “Delete” button beside the event EV123, then EV123 is deleted.

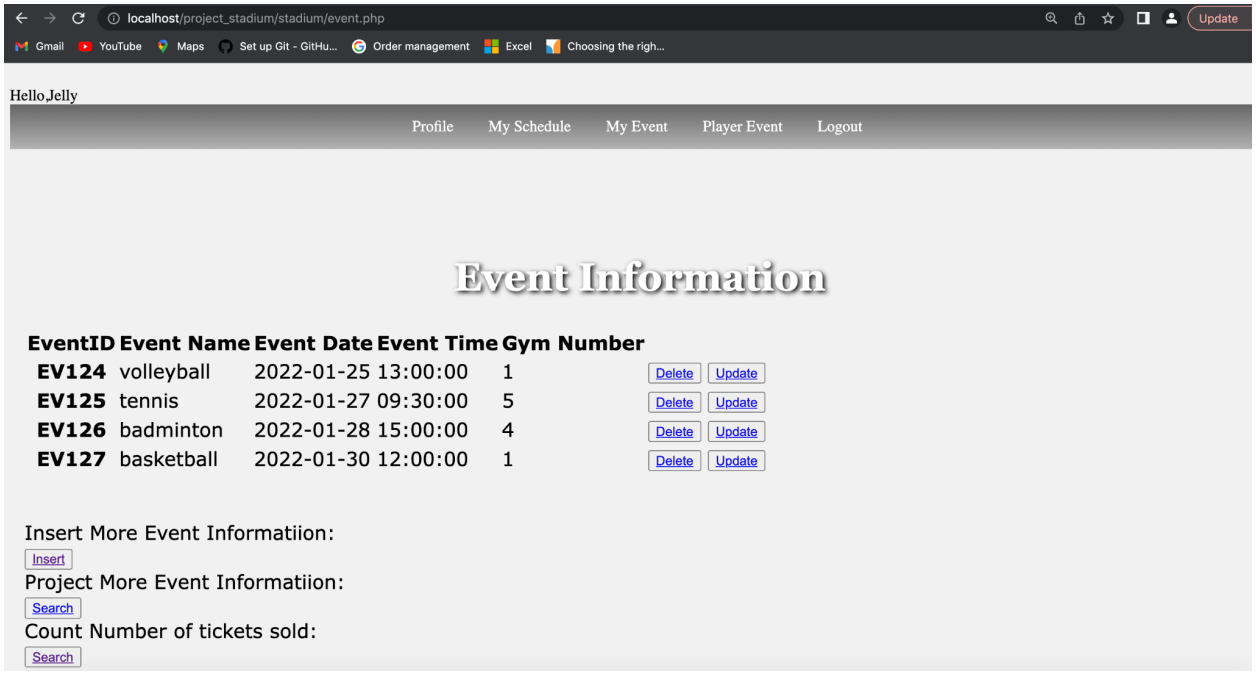

Image 2.5

Employee can click on the “Insert” button under “Insert More Event Information” to insert new event information(Image 2.6):

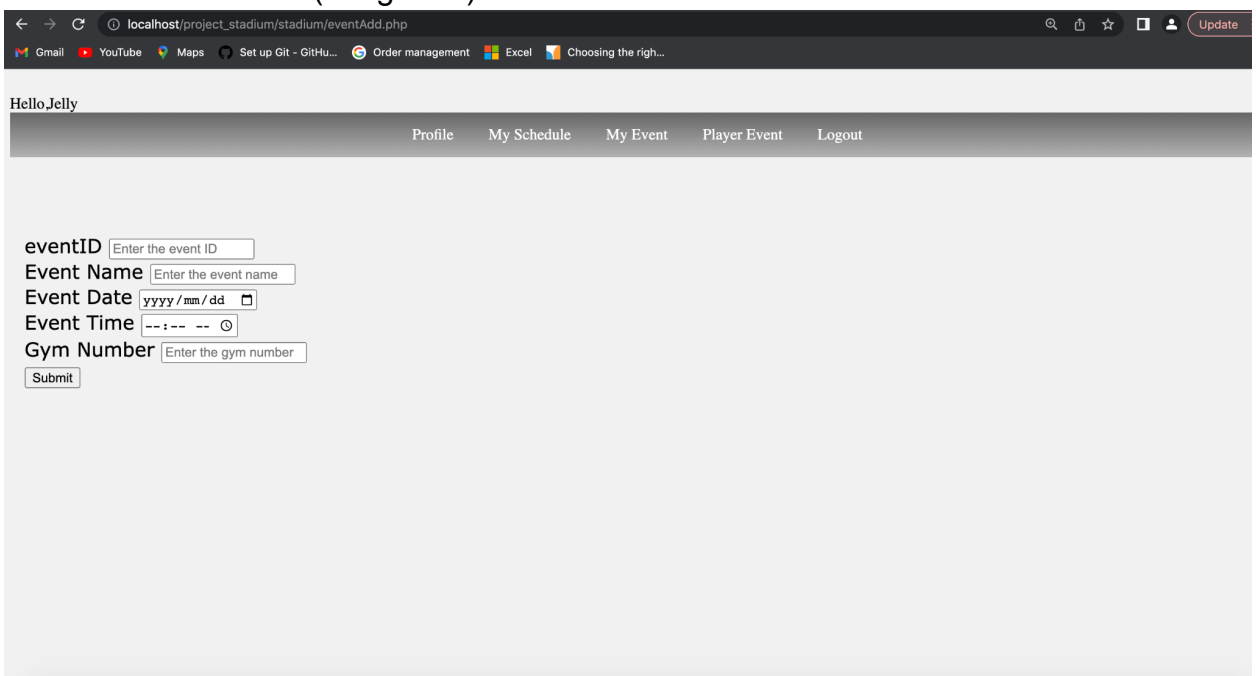

Image 2.6

Employee can fill in the new event information and click on the “Submit” button to insert new event information into database (Image 2.7):

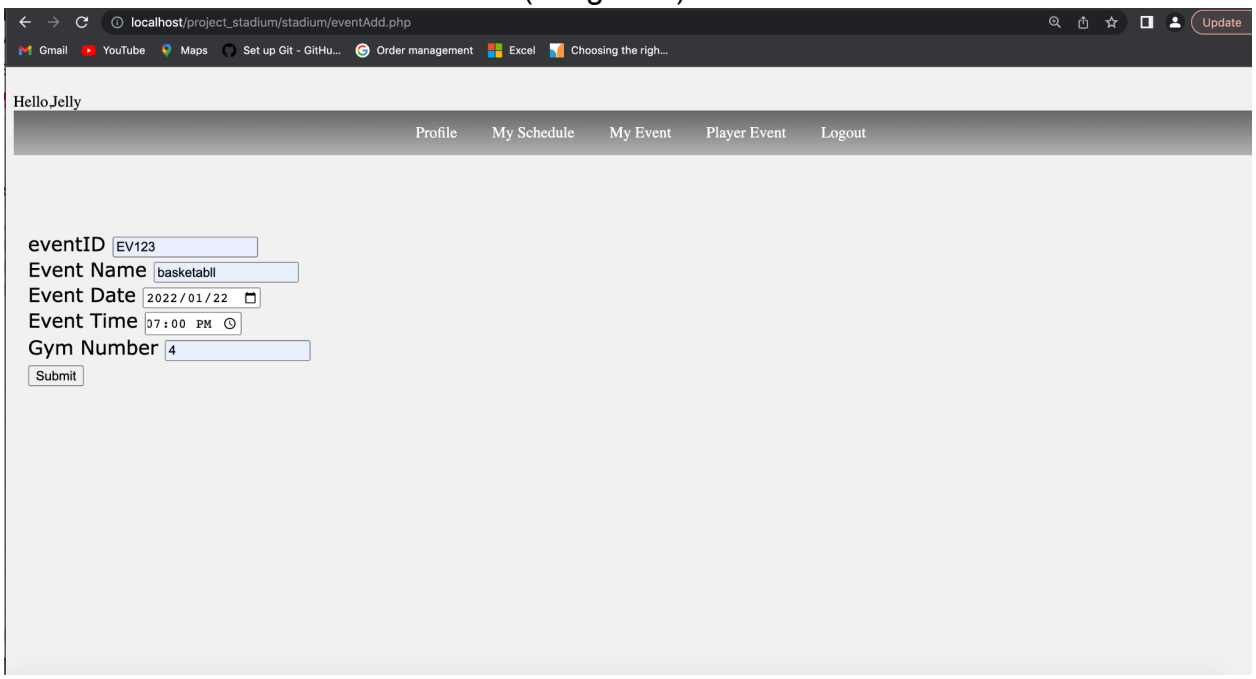

Image 2.7

The new event information is displayed in the interface(Image 2.8):

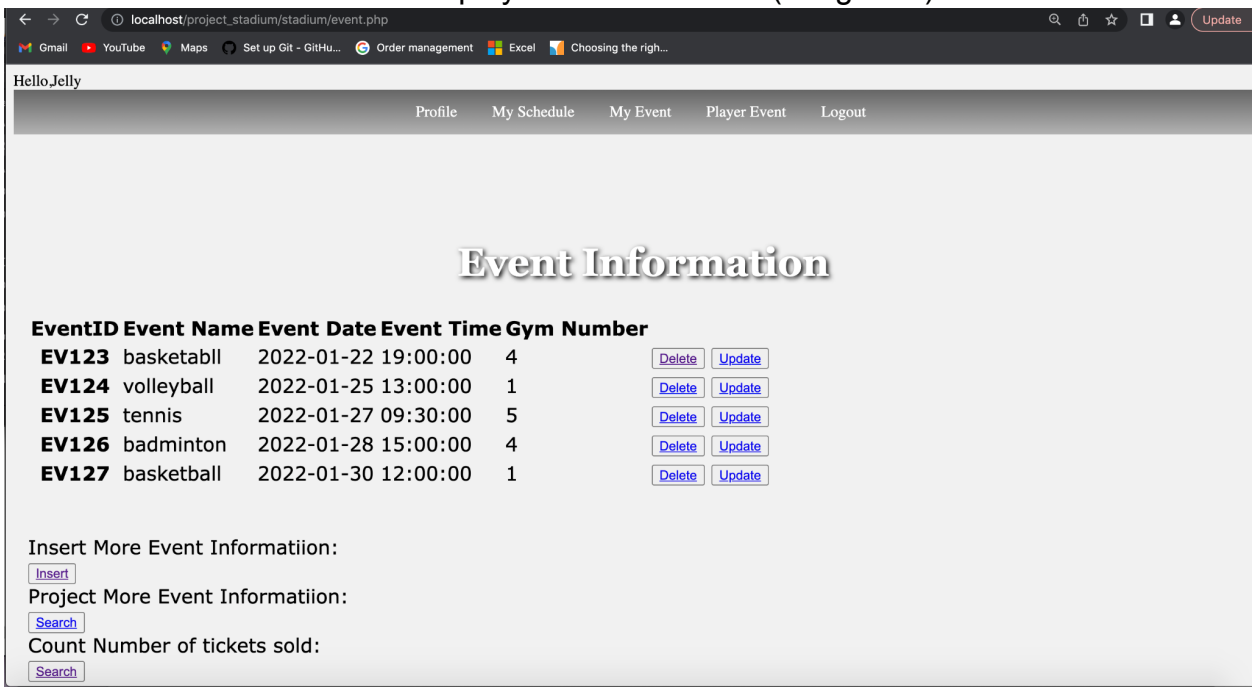

Image 2.8

Employee can also update any new information of existence event by click on the “Update” button beside each event(Image 2.9):

- For example, click on the “Update” button beside EV123, and update the event information of EV123. Here we changed the event date to “2022/04/01”.

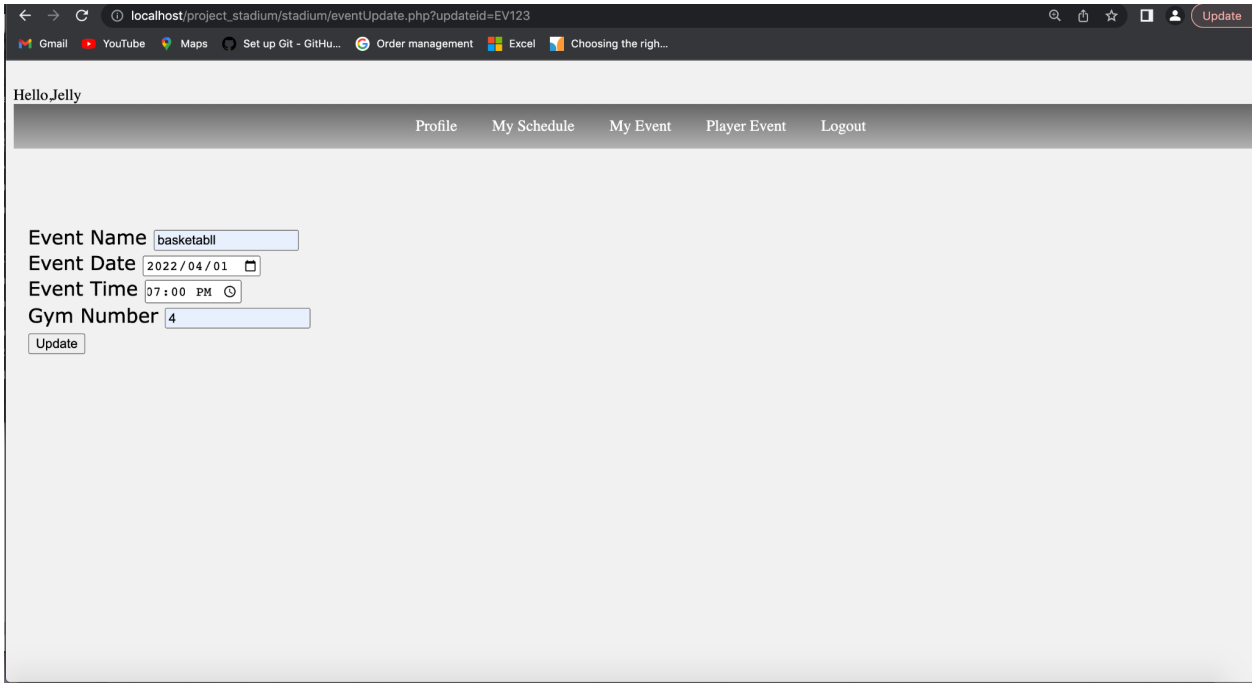

Image 2.9

The event information of EV123 is updated(Image 2.10):

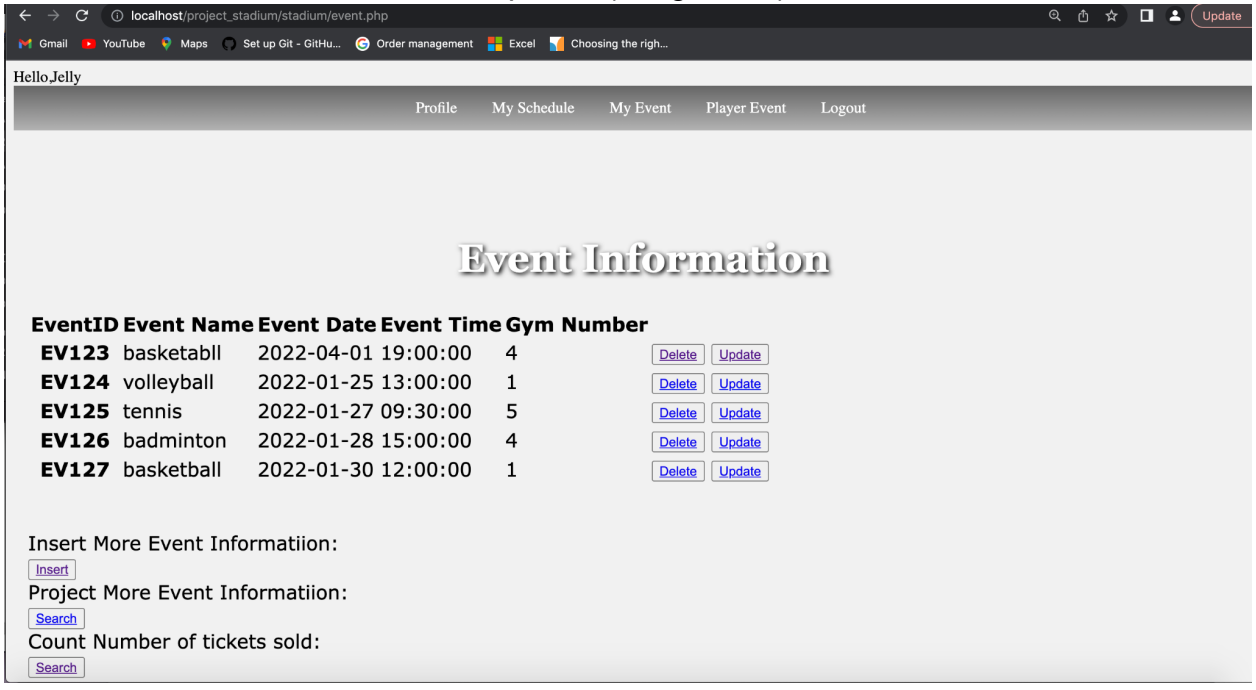

Image 2.10

Employee can also click on the “Search” button under the “Project More Information” to view all event id, all event name, all event date, all event time or all gym number(Image 2.11):

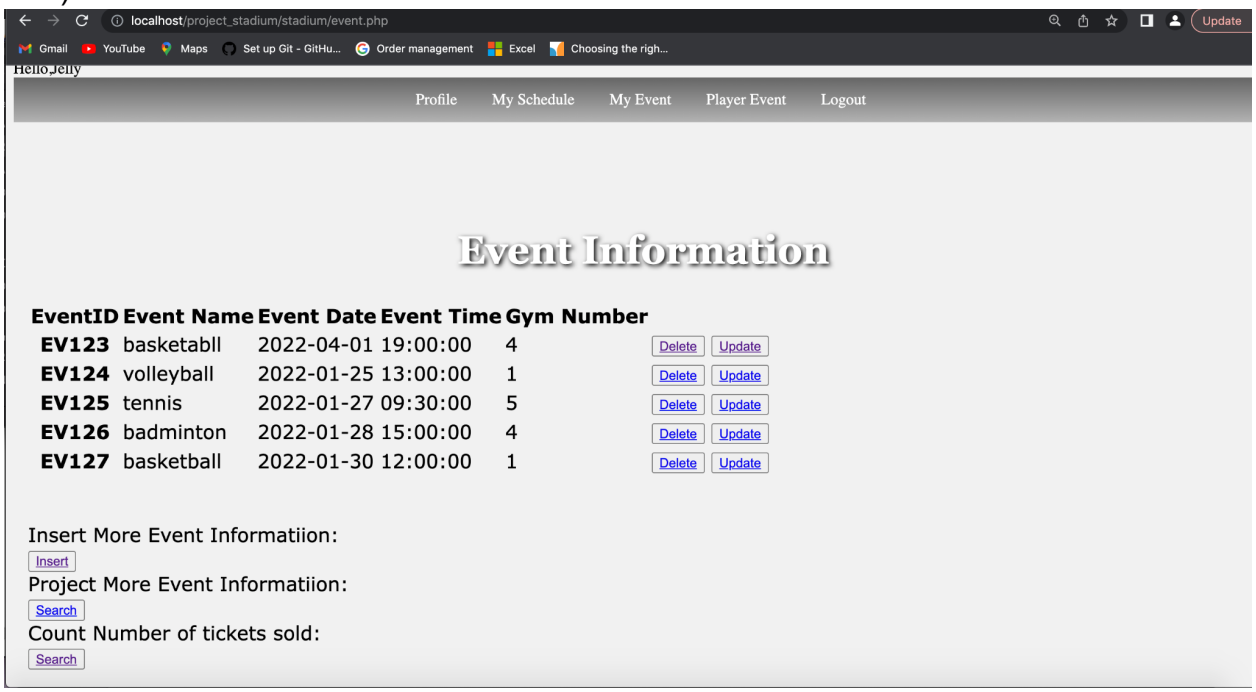

Image 2.11

Employee can click on the any given choose to view the information(Image 2.12):

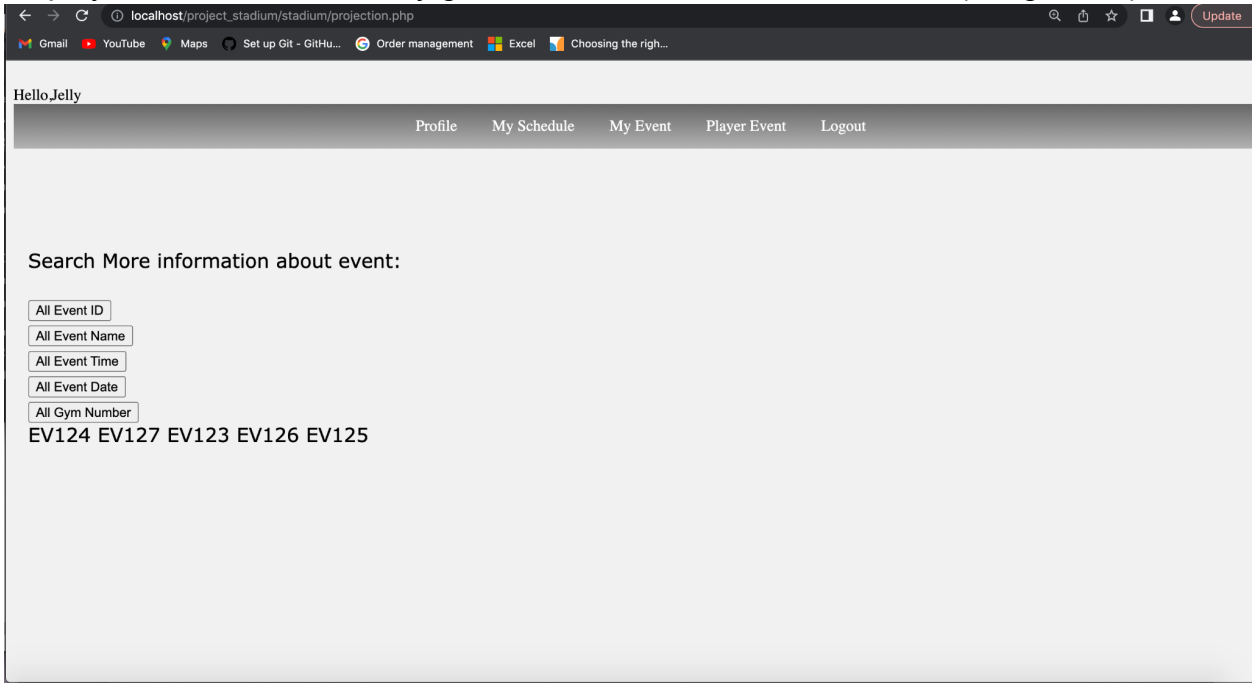

Image 2.12

For example, click on the “All Event ID” and the following data will show on the page(Image 2.13):

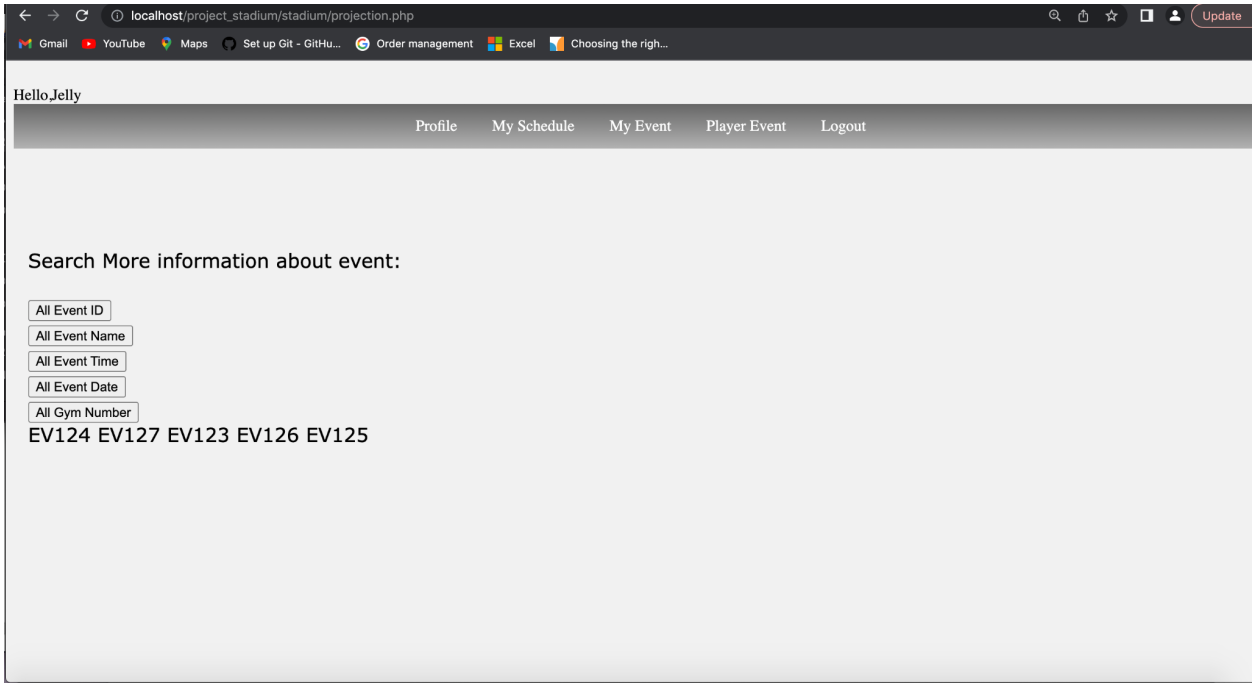

Image 2.13

Employee can also click on the “Search” button under the “Count Number of ticket sold” to view total number of ticket sold of each event(Image 2.14):

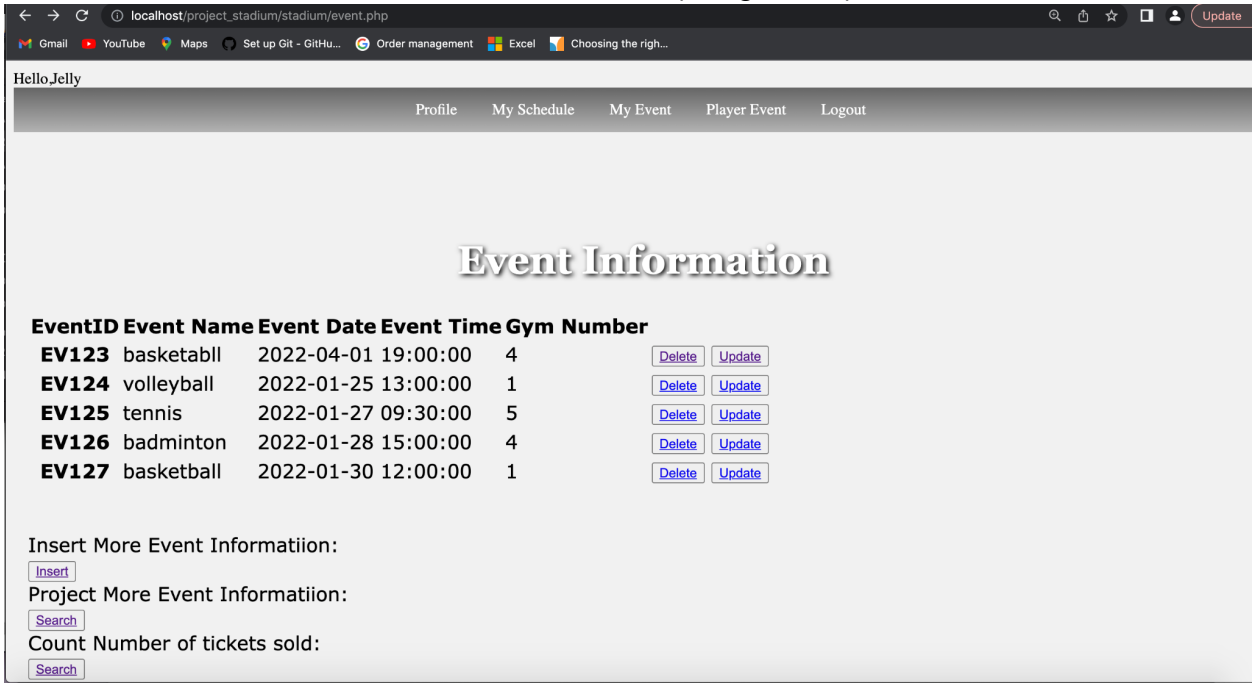

Image 2.14

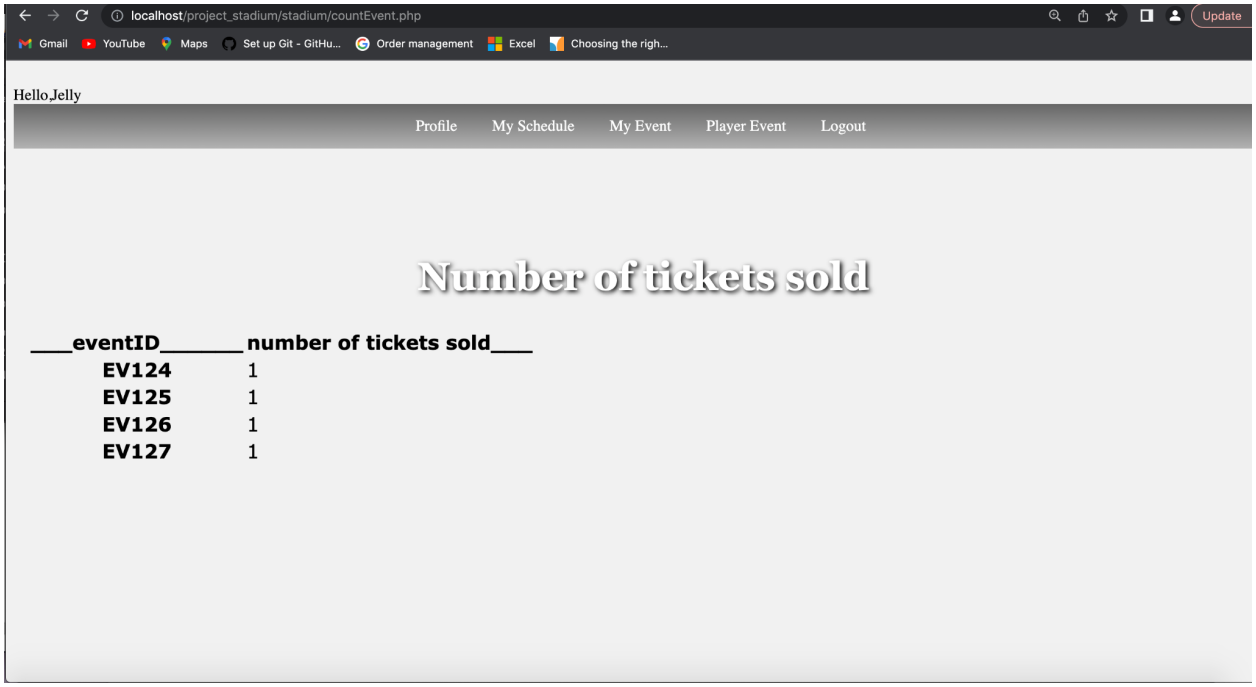

Image 2.15

Employee can also click on the “Player Event” button to view all the player information and the event they performed in(Image 2.16):

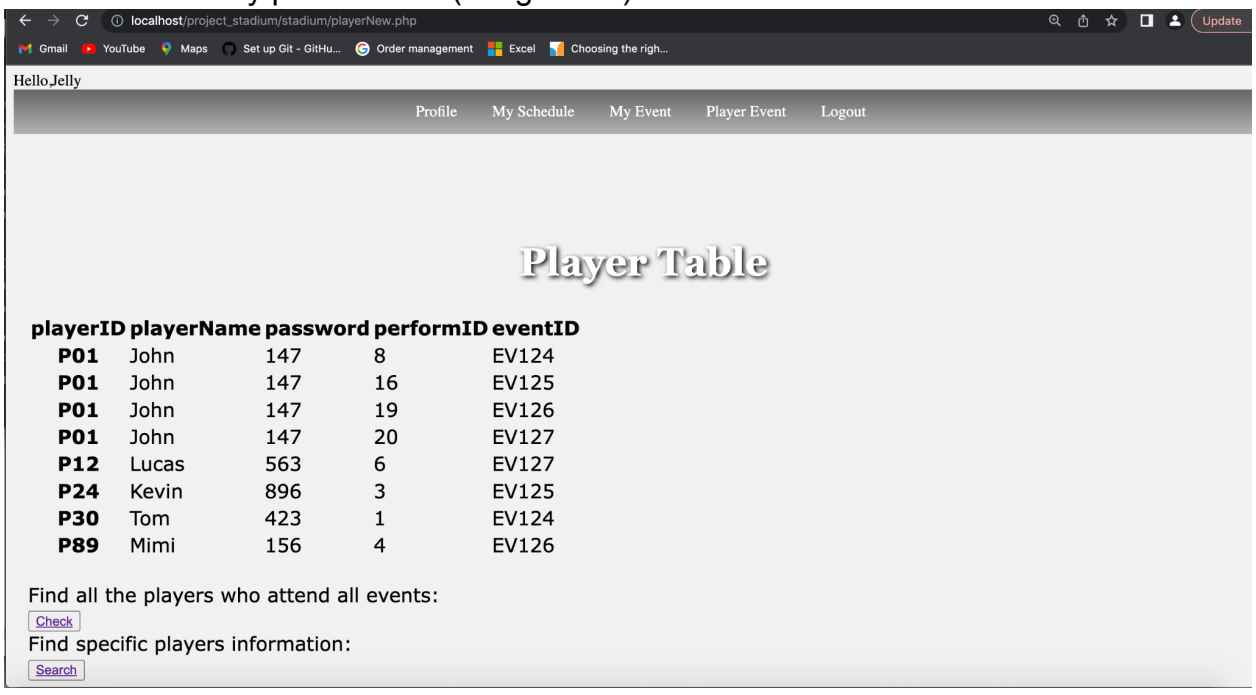

Image 2.16

Employee can click on the “Check” button under the “Find all the players who attend all events” information(Image 2.17):

- John is the only player who attends all events.

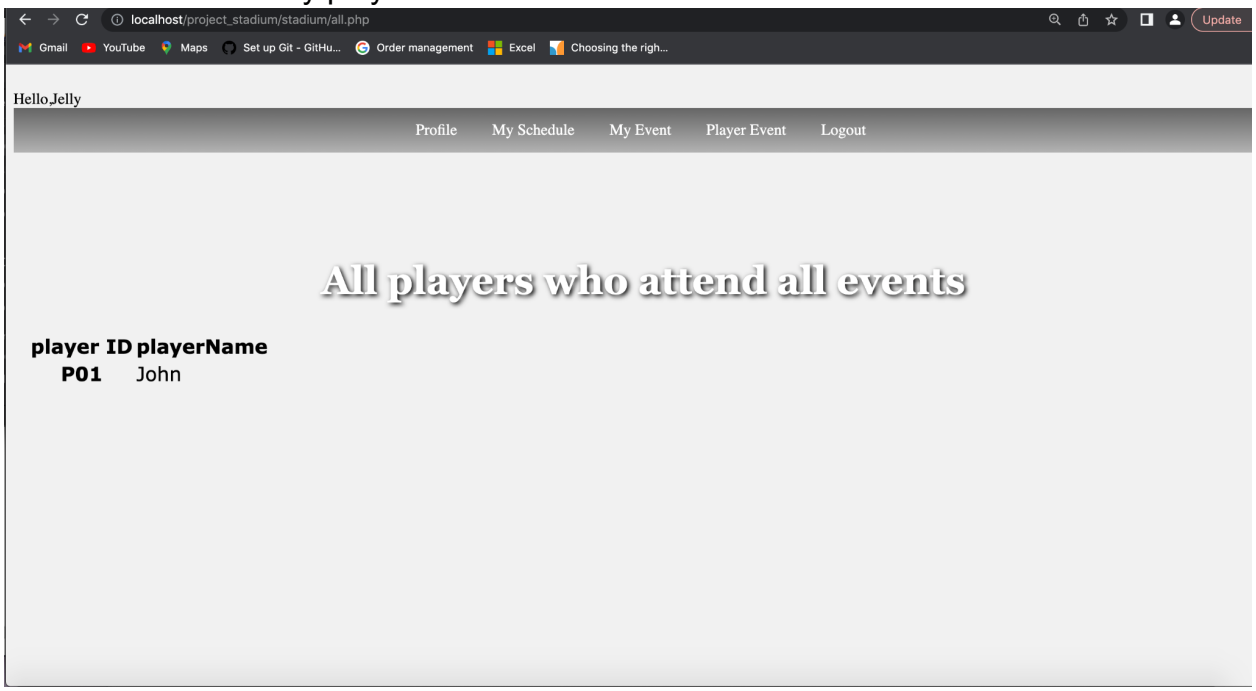

Image 2.17

Employee can also click on the “Search” button under the “Find specific players information”(Image 2.18):

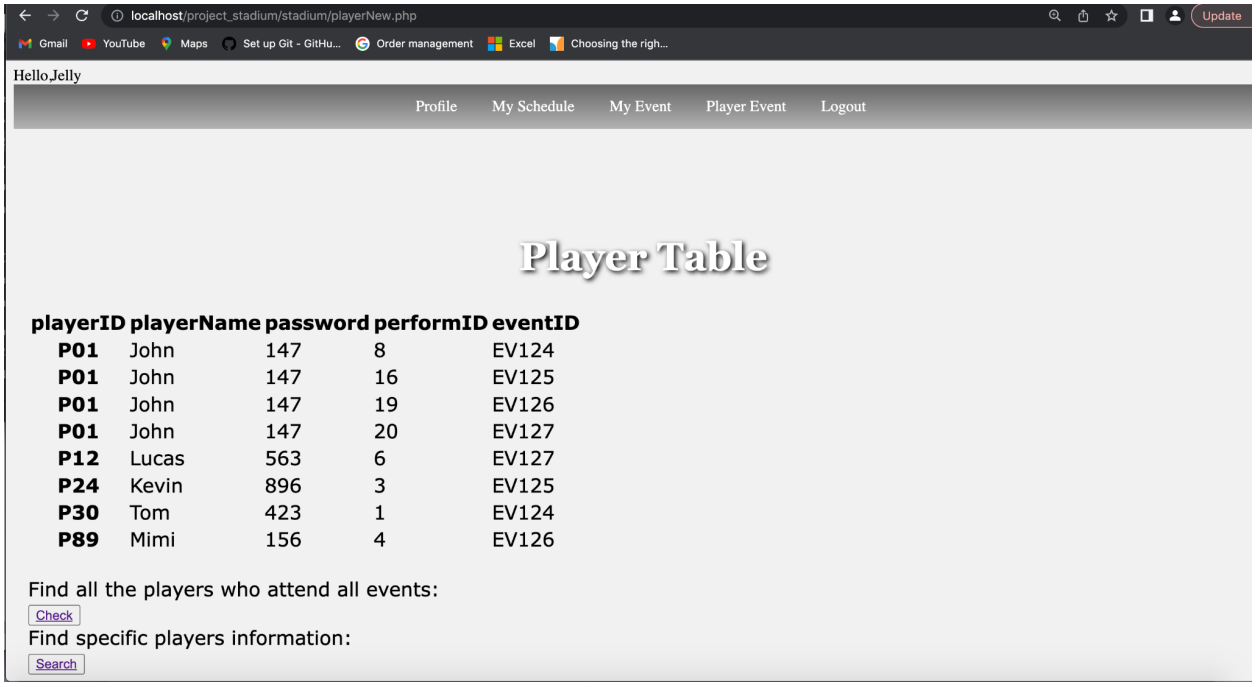

Image 2.18

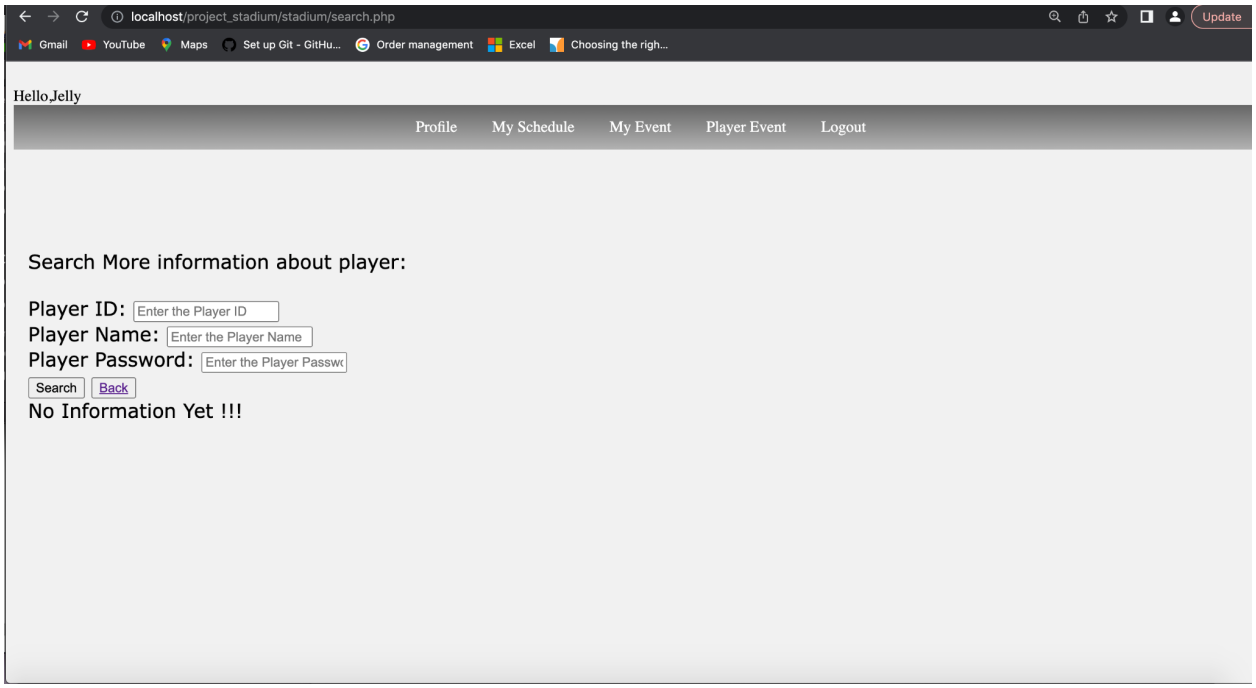

Image 2.19

Employee can fill in the player information and click on the “Search” button to view their “perform number” and “event ID”(Image 2.20):

- For example, we want to find the “perform number” and “Event ID” of player P01:

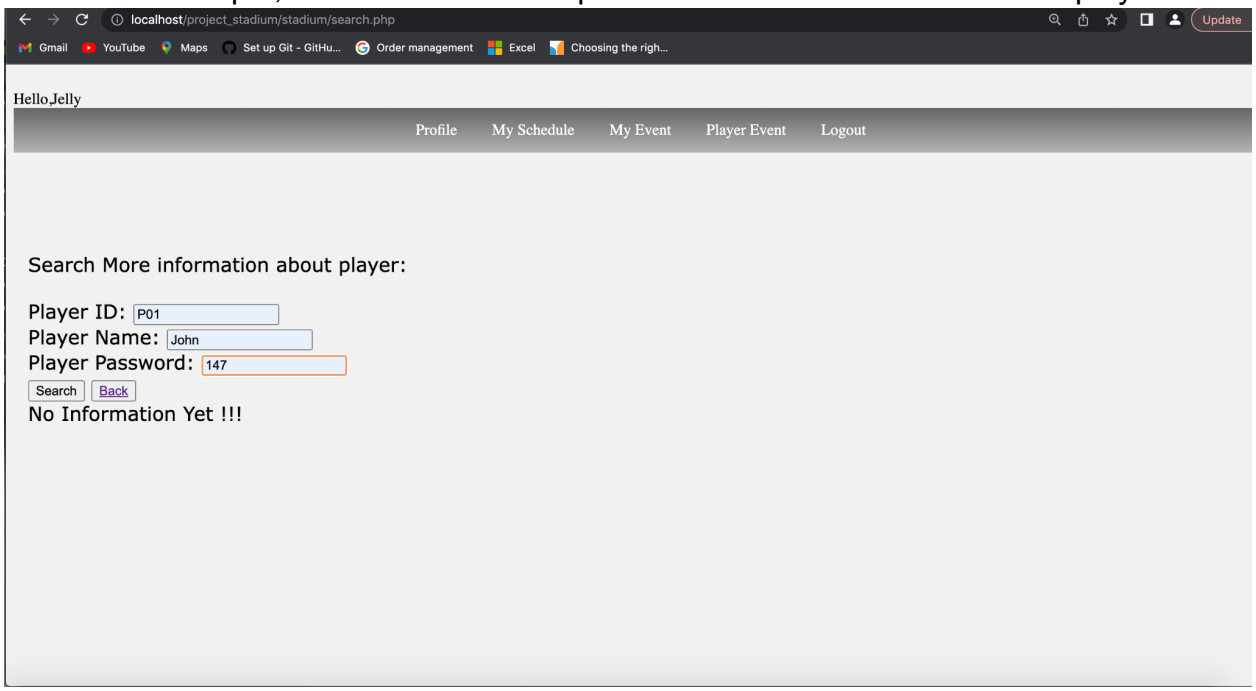

Image 2.20

The output is the following(Image 2.21):

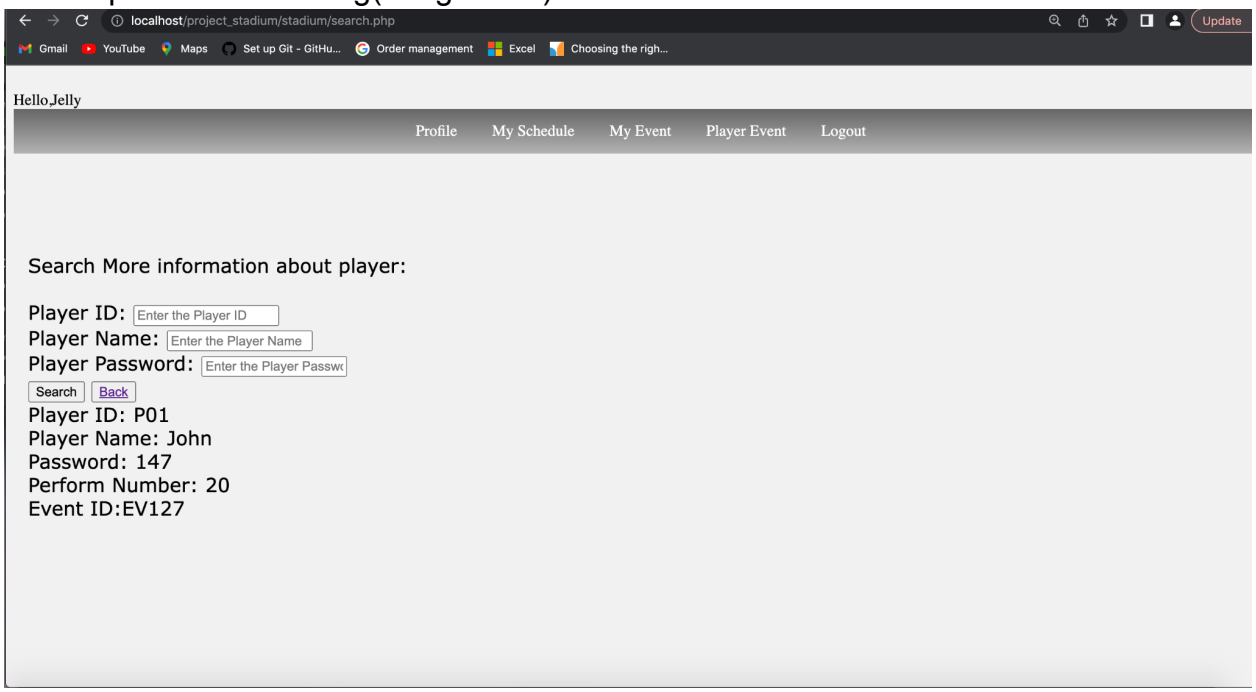

Image 2.21

Finally, the employee can click on the “Logout” button to logout the page(Image 2.22. Image 2.23):

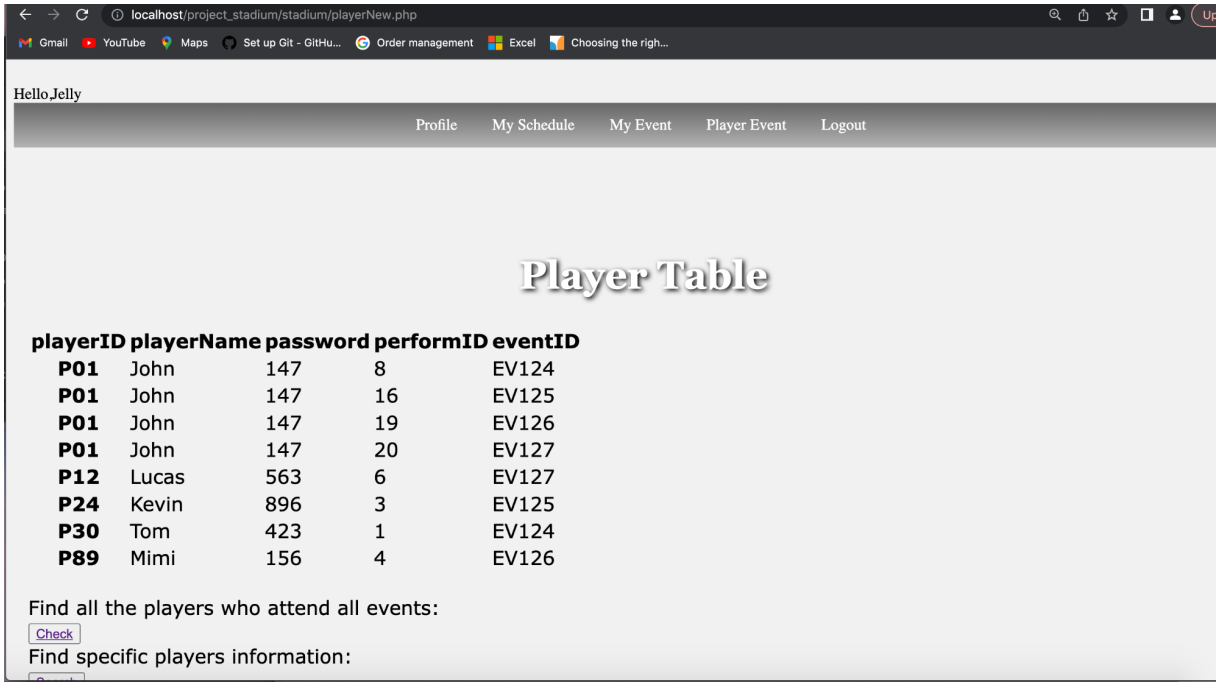

Image 2.22

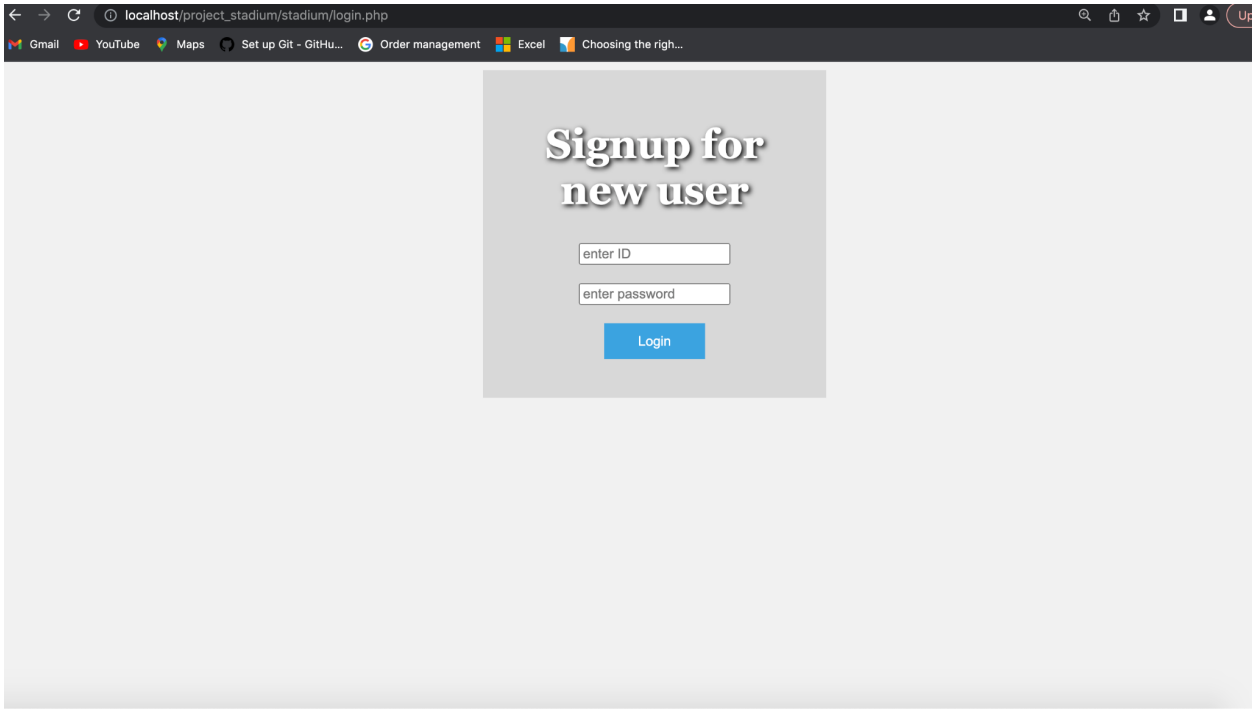

Image 2.23