10.2D: User Validation by Using Database

Tasks

In this task you are required to implement a NodeJS server that stores user details in a database for the (fictitious) *Deakin Chess Club*.

As part of this task you will need to implement a simple **Login** page. Here the data pasted from the login page can be validated against the data stored in the database on the server.

Steps

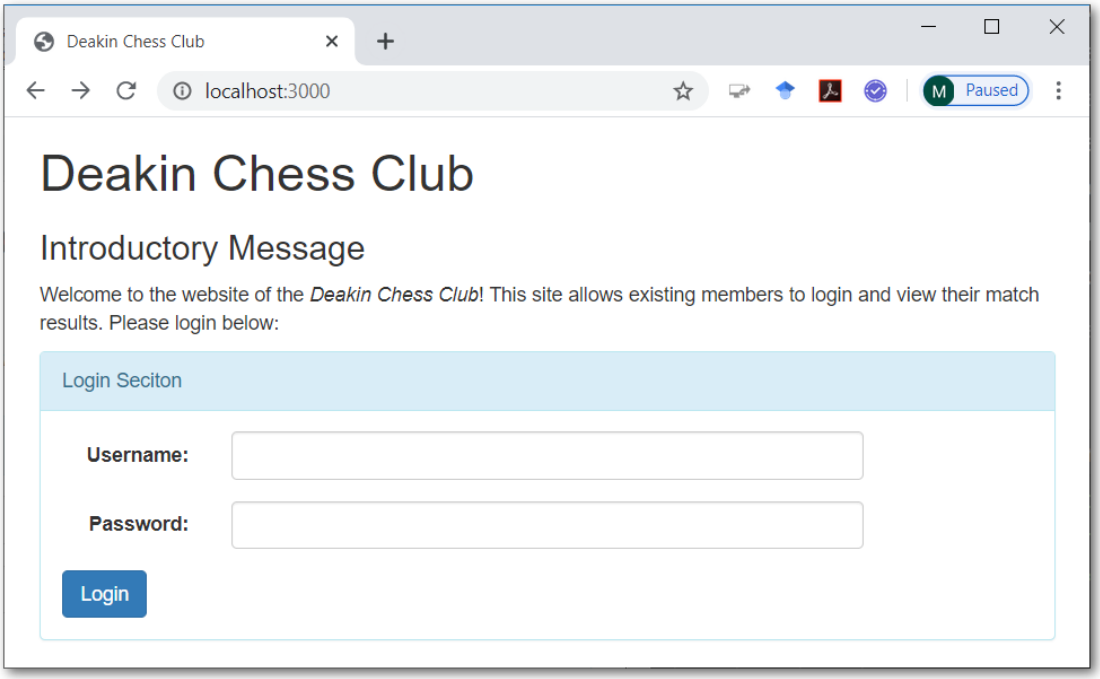
1. Create a NodeJS application (i.e., `index.js`) which on execution creates an SQLite3 database table (an *in memory* DB is fine). It should then populate this table with the details of the 5 players shown below:

ID	Username	Password	Full Name	Won	Lost	Drawn
1	mickeym	cheese123	Mickey Mouse	3	44	3
2	alfred	pm1903aus	Alfred Deakin	43	2	5
3	jane	qwerty	Jane Smith	12	35	3
4	john	brian1979	John Cleese	27	5	18
5	terry	montyp1969	Terry Jones	34	9	7

   NOTE: This table should contain 7 fields, each representing the columns in this table.

2. Next, you are to implement a **Login** form that asks the user for a `USERNAME` / `PASSWORD` combination. The user can then **POST** this data to the server on clicking a button.

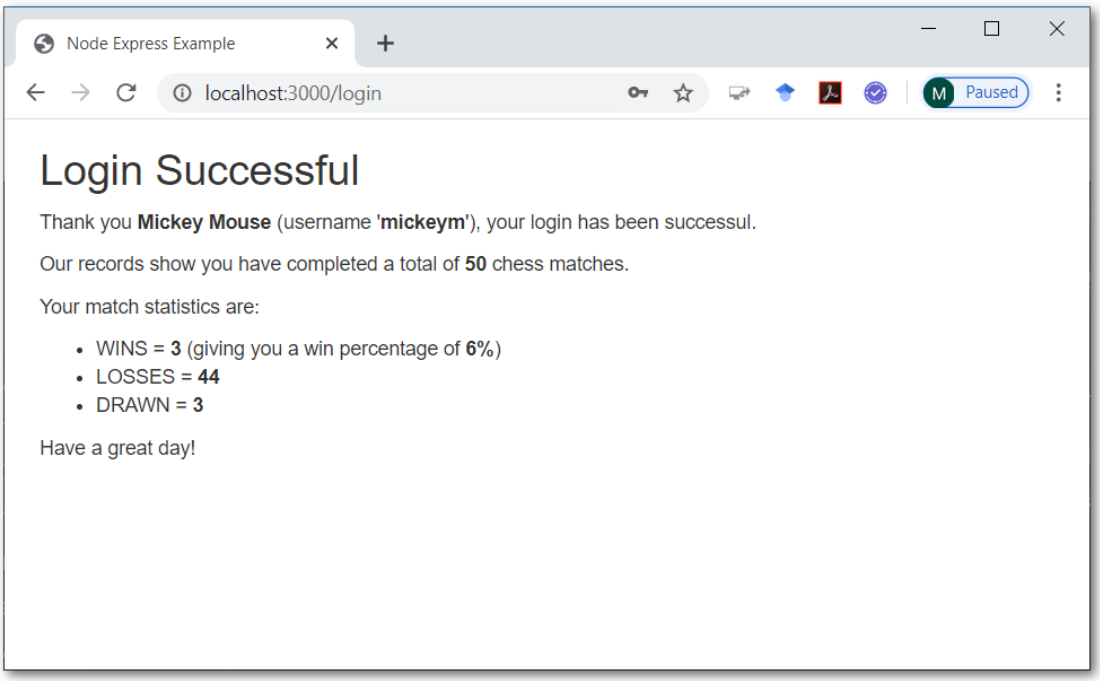
   An example of a login screen is shown below, although you are free to implement another layout/style.



Task10.2.2 Possible login page

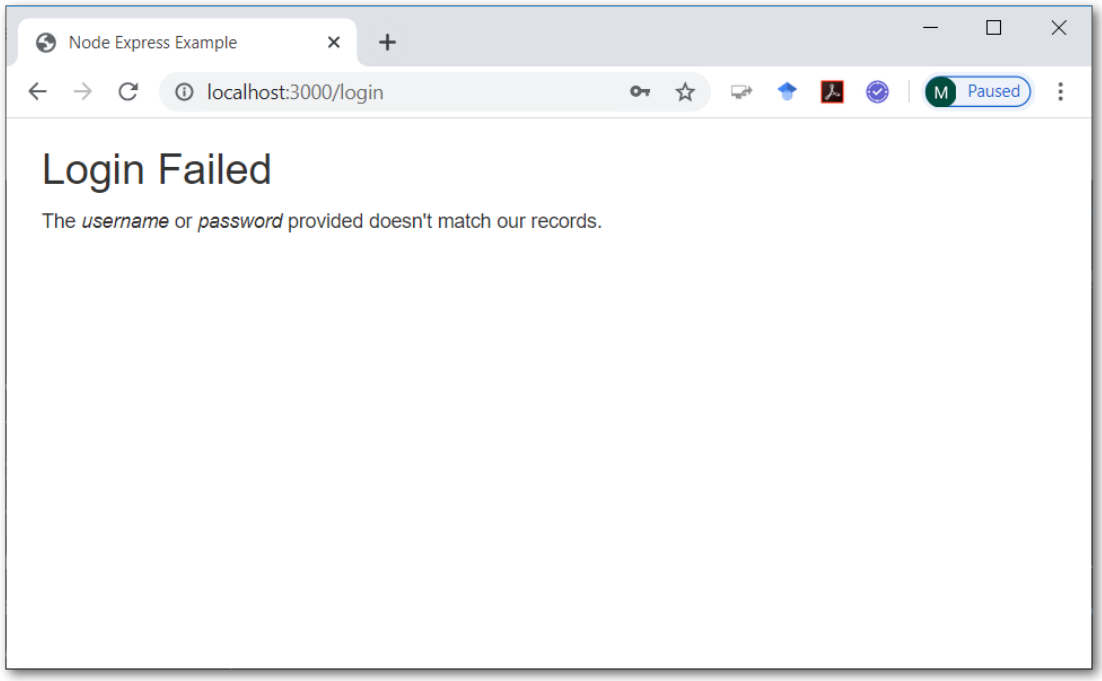
3. The NodeJS application (the server) must be able to handle the post request through an appropriate route. In processing this login request, the server should search for the given username and compare the password provided by the user with that which is stored in the database.

   If they match (login successful), an appropriate response should be provided. This should include information from the user's database record (i.e., the number of wins/losses/draws, etc.). A sample of a valid login is shown below:



Task10.2.3 Possible 'successful' login message

If the provided `USERNAME` / `PASSWORD` combination does not match the that stored in the database, an appropriate *unsuccessful* response should be provided. A sample of an unsuccessful login is shown below:



Task10.2.4 Possible 'unsuccessful' login message

Hints

- When creating the database table, consider the *data types* of respective fields, for example:
  - the the *ID* field should be an integer (see the `id INTEGER PRIMARY KEY AUTOINCREMENT` attribute in SQLite)
  - the *username* should be `UNIQUE`
  - The match results should also be `INTEGERS` as you would like to do calculations on these (to find the win %)
- When checking on the `USERNAME/PASSWORD` in the database, consider using the `SELECT` command with a `WHERE` that combines these two inputs as the search field.

What will you submit?

You should submit:

- Source code of the *login form* web page of your website.

- Source code of the *Node.js* file (i.e., the `index.js` file) that:
  - creates a server file database with a table in it
  - implements the validation of the `username/password` with that found in the DB and returns the appropriate response.
- Screenshot of the browser window showing the form web page with entered data.
- Screenshot of the browser window showing the retrieved data from the database table after a **SUCCESSFUL** login (validation) attempt.
- Screenshot of the browser window showing a response after an **UNSUCCESSFUL** login attempt.