Fall 2020 CSCI 4131 – Internet Programming
Homework Assignment 7 (Version 2, Nov 30)
Due Monday, December 14 at 11:59pm
Late Submissions, with Penalty, accepted until Tuesday 12/15 at 11:59pm

1 Description

In this assignment, you will continue to add more functionality to the website you developed in the last assignment. Your task is to upgrade the previous assignment to add a new user-administration page with functionality that enables you to add new users, delete users, or update information on existing users of your website. Adding this functionality will require you to compose and submit delete and update SQL queries to the MySQL database from your node.js/Express server with information gathered from the user via HTML pages that you will create.

You will also work with XML (Extensible Markup Language) in this assignment. To enable flexibility and increase security, the database configuration will be saved in an XML file. Your server will read the configuration information from the XML file, and that information (which is the same information that you used in assignment 6) to connect to your MySQL database.

2 Preparation and Provided Files

1. The setup for Node.js and MySQL remains the same as the last assignment.
2. The code for this assignment should be placed in a directory named: ***yourx500id_hw07***
3. The code from the last assignment can serve as a base for this assignment. Copy the code from the last assignment and place it in ***yourx500id_hw07*** directory.
4. You can use any *npm* module that require for this assignment. You should use the modules you used in previous assignments, and you will need a module for parsing XML, Thus, you might find the following *npm* module useful: ***xml2js***
5. You should determine the project structure (where you will store the various html, css and JavaScript files) for this assignment on your own.

Note: *We have provided the following sample XML file which stores database configuration:*

`sample_dbconfig.xml`

*Replace the values in the user, password, and database tags with the user/database id and numeric password that was provided via Canvas assignment (**MYSQL Database Account and Password**)*

We have also the following html file that you have permission to use as your admin page:
admin.html
Each student enrolled in this course (Csci 4131 Fall 2020) has permission to use the page for this assignment, and for their optional project for the course – make sure to adhere to the copyright statement in the file, which is displayed when the page is rendered by a browser.

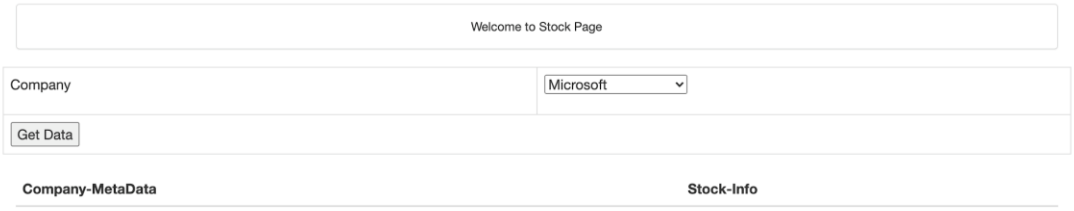
3 Functionality

Your solution should retain all functionality from Homework assignment 6, and add the functionality discussed in the following pages (build on your existing code).

The existing navigation bar should now also provide a link to the **Admin page** and **logged in User.** Check the following screenshots:

*Notice that the navigation bar in the pages listed above and shown below has a link to the **Admin page**.*

Day	Event	Time	Event Location (Virtual or Physical)	Phone Number	Extra Info (URL)
Monday	Csci 4131 Lecture	9:45 AM - 11:00 AM	Virtual (Zoom)	See Zoom Meeting info on Class	Ccsi 4131 Info
Monday	Csci 2081 Lecture	1:25 PM - 2:15 PM	Virtual (Zoom)	See Zoom Meeting info on Class	Ccsi 2081 Info
Tuesday	Virtual Office Hours	12:00 PM - 1:00 PM	Virtual (Zoom)	See Zoom Meeting info on Class	Ccsi 2081 Info

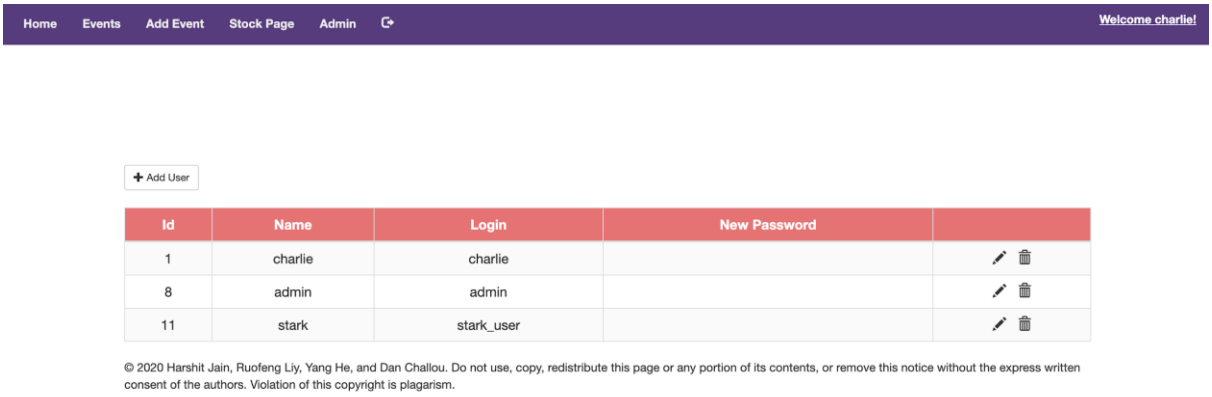


Admin page

If a user tries to access this page without a valid login, the user should be routed to the "Login" page.

- The page should have a navigation bar with a logout button.
- The table in this page should be dynamically populated with the list of users along with their corresponding Id, Name, and Login (illustrated in the following screen-shot).
- To achieve this, the server could provide a GET API which returns the list of users. The mapping between the table headers and columns in tbl_accounts is as follows:
    - Id: acc_id
    - Name: acc_name
    - Login: acc_login
    - New Password: acc_password

- The client (the page you will construct) could call the API (using AJAX) and populate the table using the data received from the server.



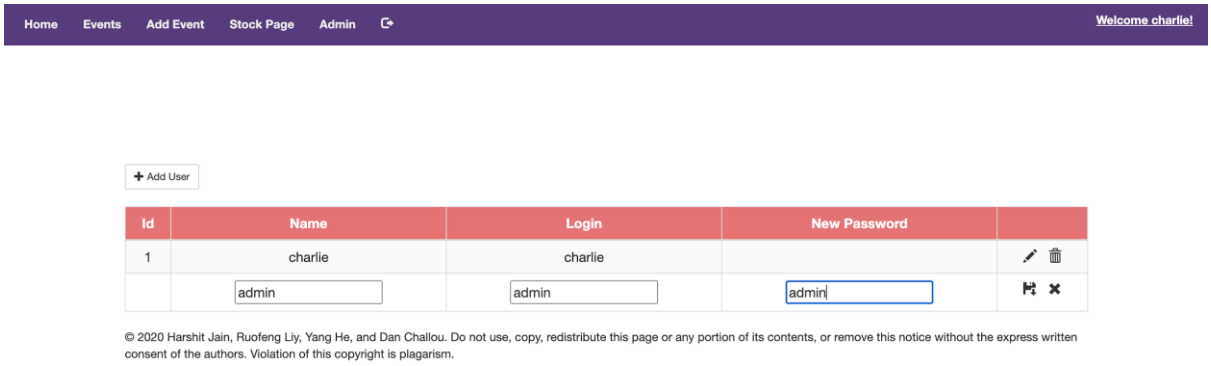
© 2020 Harshit Jain, Ruofeng Liy, Yang He, and Dan Challou. Do not use, copy, redistribute this page or any portion of its contents, or remove this notice without the express written consent of the authors. Violation of this copyright is plagarism.

Admin page - Add User functionality

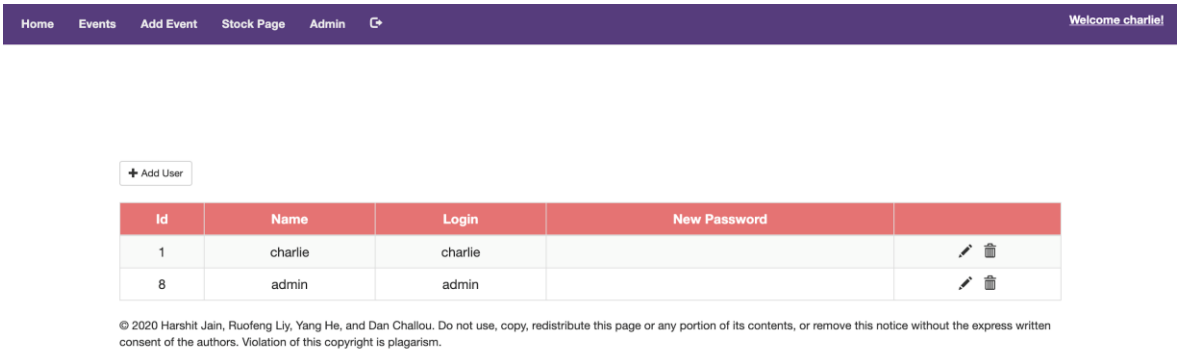
- As displayed in the picture on the above, the Admin page should have a button to add a new user.
- Upon pressing the **Add User** button, a new row with blank fields (should be added to the table) as shown in the screenshot I on the next page.
- **Name**, **Login**, and **New Password** columns of this new row should be editable. For Add User, all three fields are required. Your solution should check if these fields are empty either on the admin web page before submission and not permit submission until the fields are populated (this is what we recommend) or do so with code in your node.js/express server before saving the record to the database (if you do it on the server side, and a field is null or empty log an error to the console). Your database should not contain any user record with empty name, login, or password.
- A user has the option to enter the details of the new user and then click either **Save** or **Cancel** button.

- If the user clicks **Save**, the data entered by the user should be posted to the server.
    - The server should validate that no other user in the database has the same login.
    - In case the validation passes, the information about the new user should be inserted in the following table: ***tbl_accounts***.
    - The new user should also be added to the HTML table.
    - In case the validation fails, the following error message should be displayed: This login is used by another user
- If user clicks **Cancel**, the new row should be removed and display should revert back to the state before **Add User** button was pressed
- **Note**: Review screenshots I. and II below for an indication of how the **Add User** functionality should operate.

I. *Admin enters the details of a new user and clicks the **Save button***



II. *The new user is successfully added.*

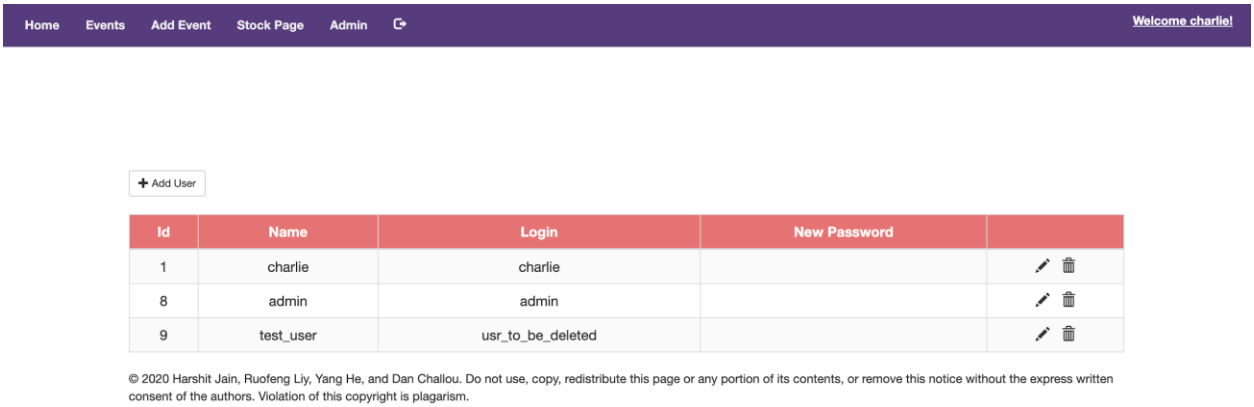


Admin page - Delete User functionality

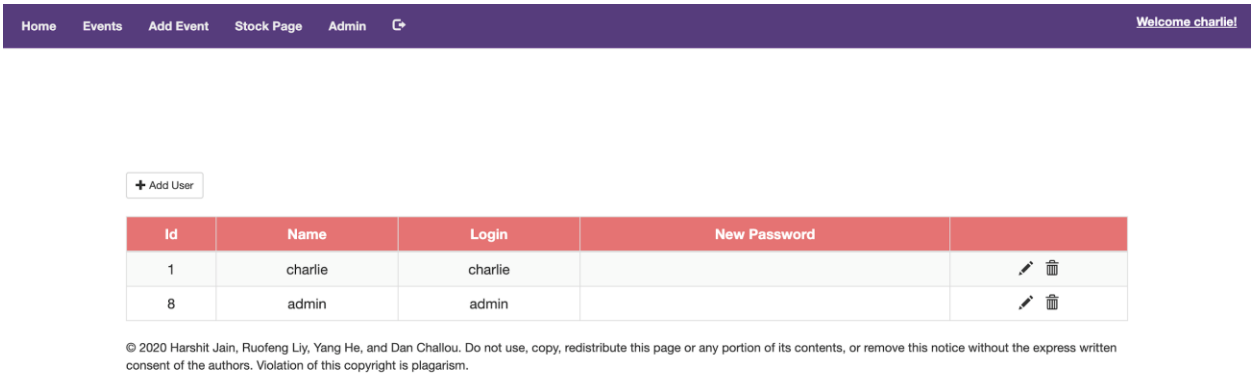
- Each row in the table should have a **Delete** button (icon) associated with it (see the garbage can in the rightmost column).
- The **Delete** button should delete the user from the database and from the HTML table only if the user being deleted is not the one currently logged in. To achieve this, the server should provide a delete API which can be used by the client.
- The server should not allow the user to delete the currently logged user. The following error message should be displayed on the Admin page in the event that a user attempts to do this: Error : Cannot delete logged in User!

- **Note**: Review screenshots A, B, and C on the next page depicting the **Delete User** functionality.

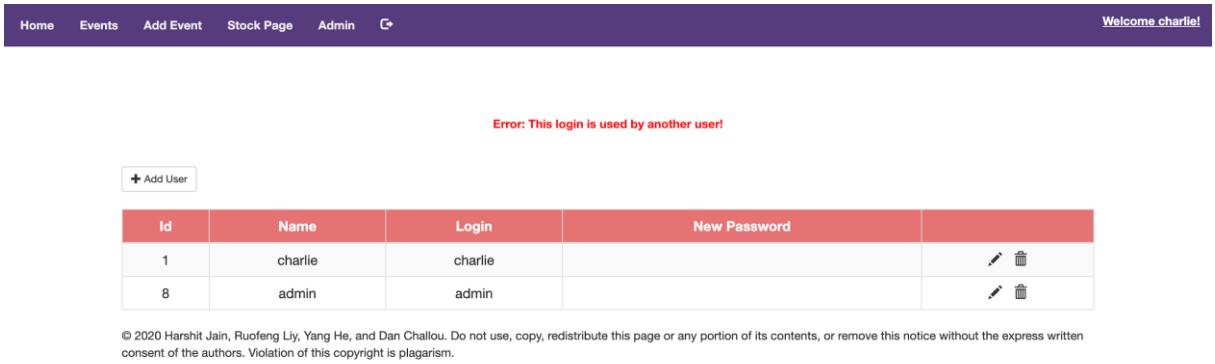
*A. Admin clicks on the delete button (trash icon) next to the user **usr_to_be_deleted**.*



*B. **usr_to_be_deleted** is deleted from the database and from the HTML table.*



*C. Admin tries to delete the user **charlie** which is currently logged in. The user is not deleted and an error message is displayed.*

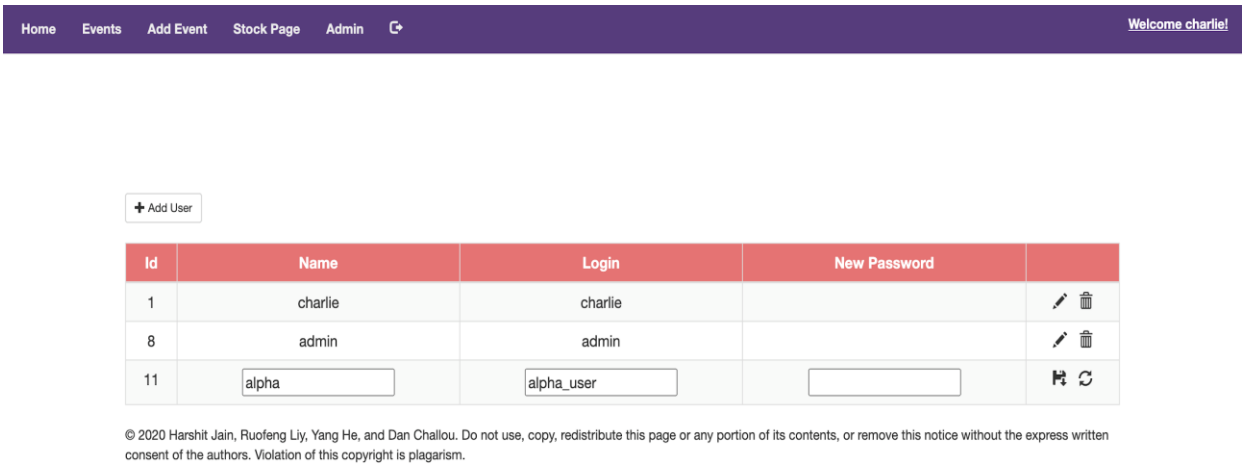


Admin page - Edit User functionality

- Each row in the table should have an **Edit** button associated with it (see the pencil icon in the just to the left of the garbage can).
- Clicking **Edit** button should activate edit mode and display it in the list of users table.
- You should be able to update the **Name**, **Login**, and **New Password** in edit mode. Your database should not contain records with any empty fields. However, in update mode, , you should allow an empty password field to be submitted to your node.js/express server. In update mode, and empty password field implies that only the name and login may should be changed (and the existing password should not be changed). You should identify such cases on the server side to ensure an empty password (i.e., "") is not saved to the database.
- Edit mode will have two different buttons: **Update** and **Cancel**.
- **Cancel** should **discard** the changes and exit edit mode.
- Upon clicking the **Update** button, the data entered by the user should be posted to the server.
    - The server should validate that no other user in the database (other than the one being edited) has the same login.
    - In case the validation passes, the information about the edited user should be updated in the following table: ***tbl_accounts***. The updated information should also reflect in the HTML table.
    - In case the validation fails, the following error message should be displayed: Error: This login is used by another user.

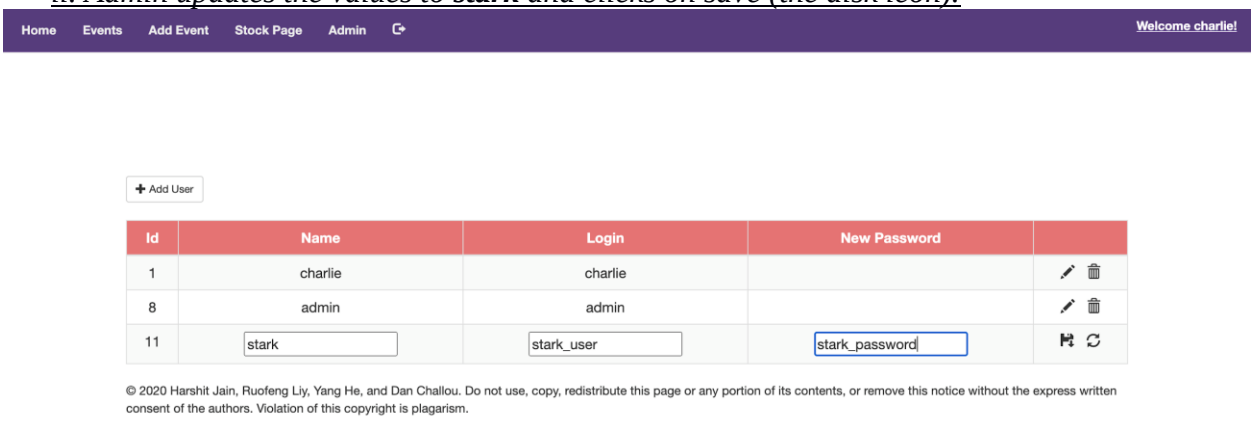
Note: Review the screenshots i, ii, ii, iv, and v below and on the next 2 pages depicting the **Edit User** functionality.

i) *Admin clicks on the edit button next to the user **alpha**.*

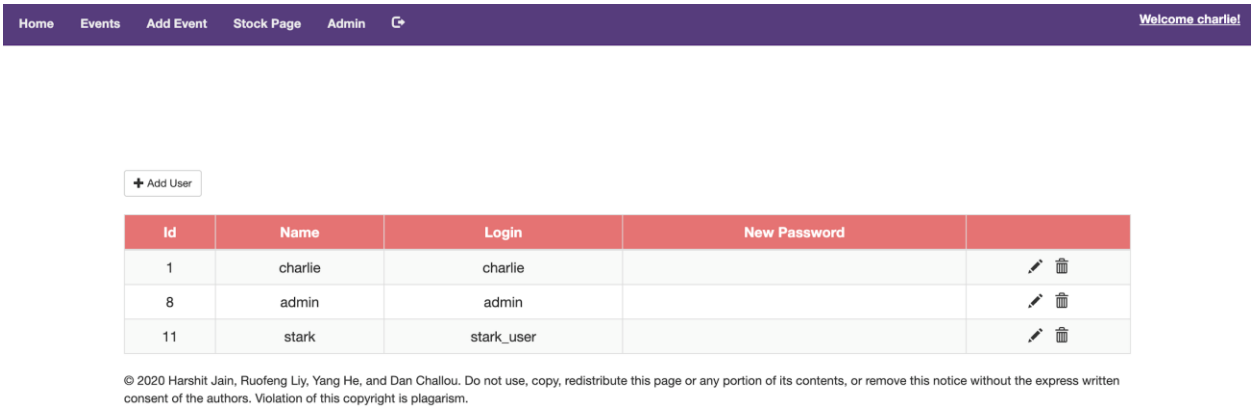


© 2020 Harshit Jain, Ruofeng Liy, Yang He, and Dan Challou. Do not use, copy, redistribute this page or any portion of its contents, or remove this notice without the express written consent of the authors. Violation of this copyright is plagarism.

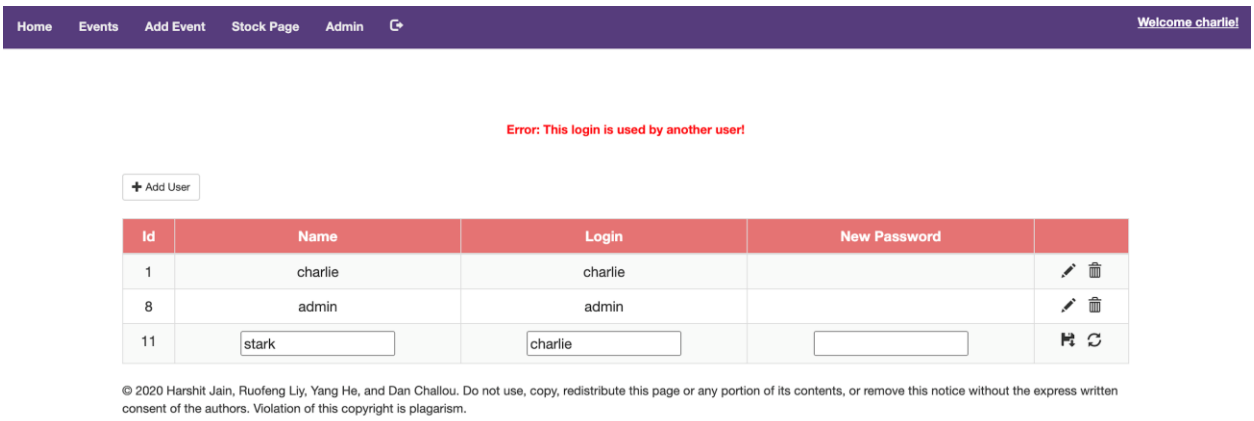
*ii. Admin updates the values to **stark** and clicks on save (the disk icon).*



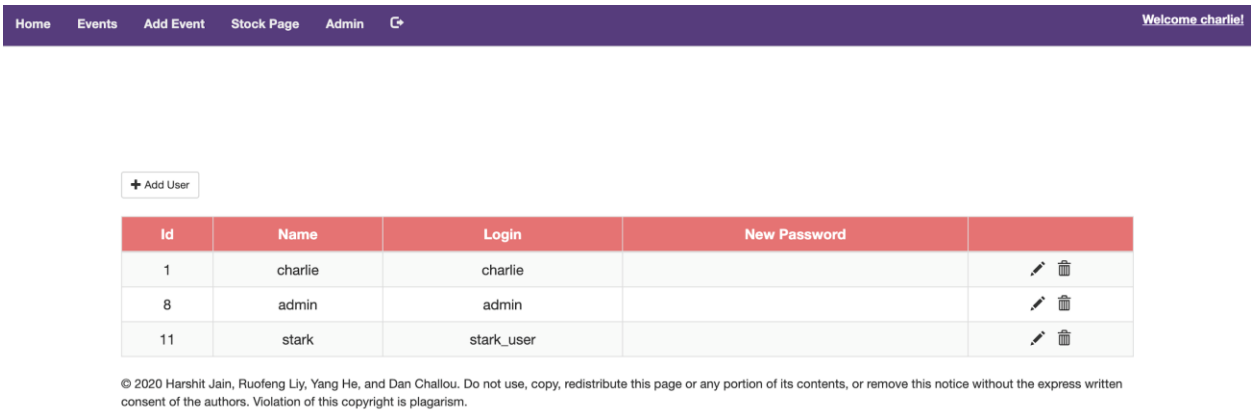
ii) The values are successfully updated in database and displayed in the user table.



*iii) The user edits the details of user **stark** and updates the login to **charlie.** Error message is displayed because a user with login **charlie** already exists.*



iv) *Admin clicks on **Cancel** button to discard the changes (the circular icon next to the disk icon).*



Navigation bar

- The navigation bar displays the user which is currently logged in.

Database Configuration

- Your server should read the values in the file: `sample_dbconfig.xml` (***which was provided, and YOU MUST UPDATE***) and use them to establish a database connection. Note, you can only run the parser from node.js/Express, so make sure to insert ample calls to console.log to check to make sure your server is functioning as you expected as you develop your solution.

4 Submission Instructions

PLEASE ENSURE TO TEST YOUR CODE ON CSE LAB MACHINES.

You will need to submit all the files used to develop the website. This includes all the HTML, CSS, JavaScript, package.json, index.js and any other files.

** **Make sure to add the user information below to your accounts table (tbl_accounts). Violation to this will result in penalty. (-10 points, at a minimum)**

Login: admin
Password: admin

We will use this login and password values to login to your website. Your assignment should be submitted in a single compressed file with the name ***yourx500id_hw07*.** The server file should be named as ***index.js***.

5 Evaluation

Your submission will be graded out of 100 points on the following items:

- **Submission instructions are met** (5 points).
- Navigation bar in all the pages displays the currently logged user. (5 points)
- Admin page displays the correct list of users in the system. The Admin page should display at least two other users besides the admin user. (10 points)
- In the Admin page, the DELETE button works for every row and displays error messages correctly. (10 points)
- In the Admin page, the EDIT button works for every row and switches the view to EDIT mode. (10 points)
- In the Admin page, the UPDATE button works in EDIT mode. It validates the input, and updates the name, login, password accordingly. All the test cases should work (10 points)
- In the Admin page, the CANCEL button works in EDIT mode. (5 points)
- In the Admin page, ADD USER works and switches the view to ADD mode. (10 points)
- In the Admin page, the SAVE button works in ADD mode. It validates the input, and saves the name, login, password accordingly. (10 points)
- In the Admin page, the CANCEL button works in ADD mode. (5 points)
- Server reads database configuration from XML file and establishes connection to database. (10 points)
- All functionality from assignment 6 works. (10 points)

6 Assignment Guidelines and Penalties – we reserve the right to grade subjectively.

1. The illustrations in the lab assignments and demos, though not stated, are implied requirements - though the style you use can differ, it shouldn't cause your output to look worse than the example given. **Therefore, though the CSS and page elements can vary but they should function the same as illustrated in the assignment write-up and demos**, and by our judgment, they should not negatively impact the functionality described in the write-up and demos.

2. Unless stated otherwise, for assignments that build on other assignments, the functionality developed in the previous assignment(s) **should** still work correctly.

You should follow these guidelines while working on your assignments. Failure to do so may result in penalties assessed against your score that are not explicitly stated in the rubric.