SFWRTECH 4WP3 – Advanced Web Programming - Assignment #7

Weight

4% of total course grade

Due date

Monday March 21st at 11:59pm

Primary learning objectives

- Create a multipage application with NodeJS, mustache partials and the MVC pattern
- Create an application with sessions and user authentication using NodeJS
- Use middleware to implement functionality with NodeJS

Requirements

Unzip the starter code on Avenue to Learn. Run **npm install** to install the required packages. Run **node dbinit.js** to initialize the database, and then **node app.js** to start the application.

The application allows "public users" (i.e. users that are not logged in) to view articles and will allow logged-in users to create articles. You can login with username **bob** and password **test**.

Your assignment is to extend the functionality of this application in the following ways.

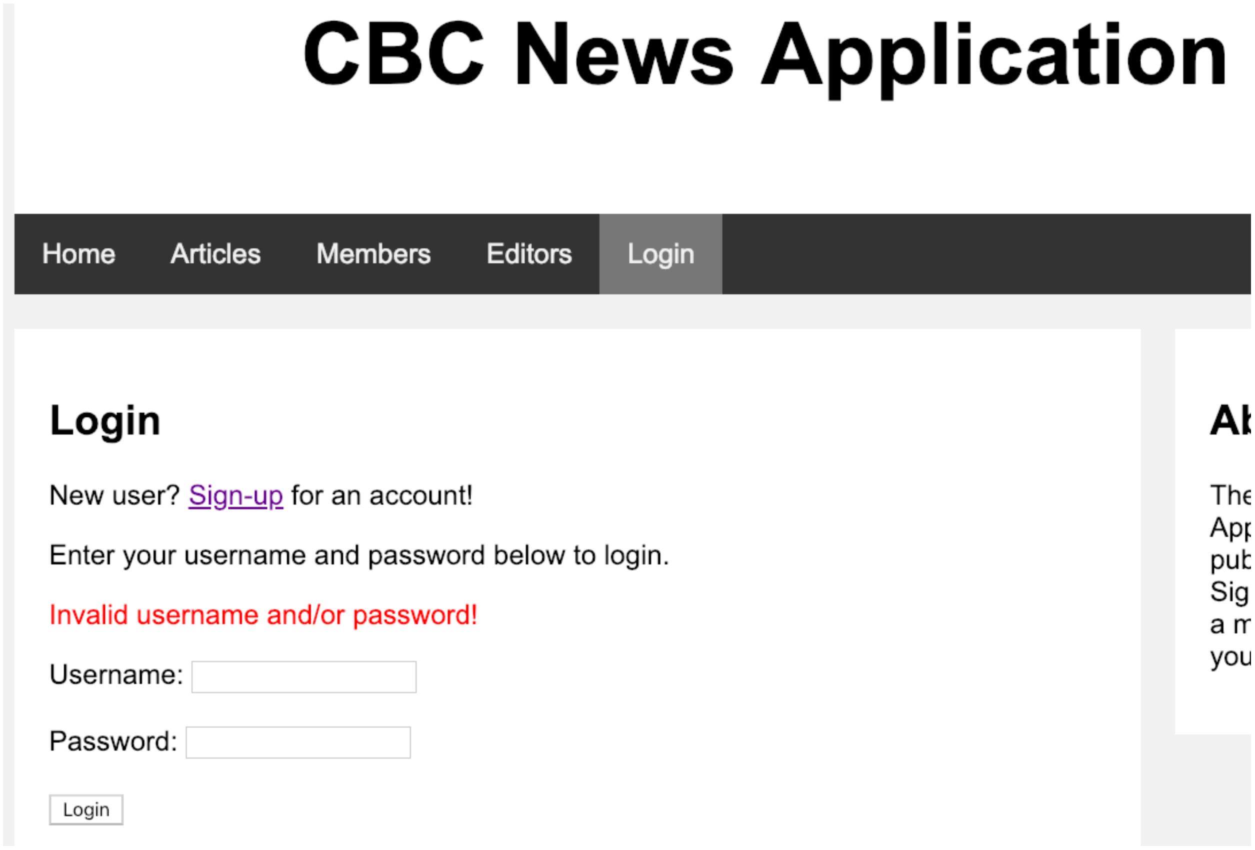
1. **Log middleware**
   - Create a middleware that upon every request to the server writes a comma-separated line to a file called **log.txt** contained in the same folder as the **app.js** script. Each comma-separated line should contain the following data:
     1) The Date and Time of the request
        - *Hint, try:* ***new Date()***
     2) The path of the request (e.g. /members).
        - *Hint, try:* ***req.path***
     3) The IP address of the requester (it will look like ::1 if running locally)
        - *Hint, try:* ***req.ip***
     4) The query parameters
        - *Hint, try:* ***JSON.stringify(req.query)***
     5) The request body (i.e. may include POST parameters)
        - *Hint, try:* ***JSON.stringify(req.body)***

```
Fri Feb 9 2018 16:10:03 GMT-0500 (Eastern Standard Time),/home,::1,{},{}
Fri Feb 9 2018 16:10:04 GMT-0500 (Eastern Standard Time),/members,::1,{},{}
Fri Feb 9 2018 16:10:04 GMT-0500 (Eastern Standard Time),/home,::1,{},{}
Fri Feb 9 2018 16:10:04 GMT-0500 (Eastern Standard Time),/articles,::1,{},{}
Fri Feb 9 2018 16:10:05 GMT-0500 (Eastern Standard Time),/members,::1,{},{}
Fri Feb 9 2018 16:10:05 GMT-0500 (Eastern Standard Time),/home,::1,{},{}
```

Figure 1 - Example Log File Contents

2. Login

- Currently a user is logged in based on a hardcoded username/password. The Users table created by **dbinit.js** is a table of usernames, passwords and access levels. Instead of using the hardcoded values, check this table for a matching username and password. If a matching username and password exists, log that specific user into the application and re-direct them to the Members page if they are a member-level user or the Editors page if they are an editor-level user. If the user does not exist in the table, reload the login page with the login error displayed ("Invalid username and/or password!"). When a logged-in user creates an article, their username should be the author of the article (currently it is hardcoded to be "bob"). Create a new Users model to help implement this functionality and other functionality involving the Users database table.



New user? Sign-up for an account!

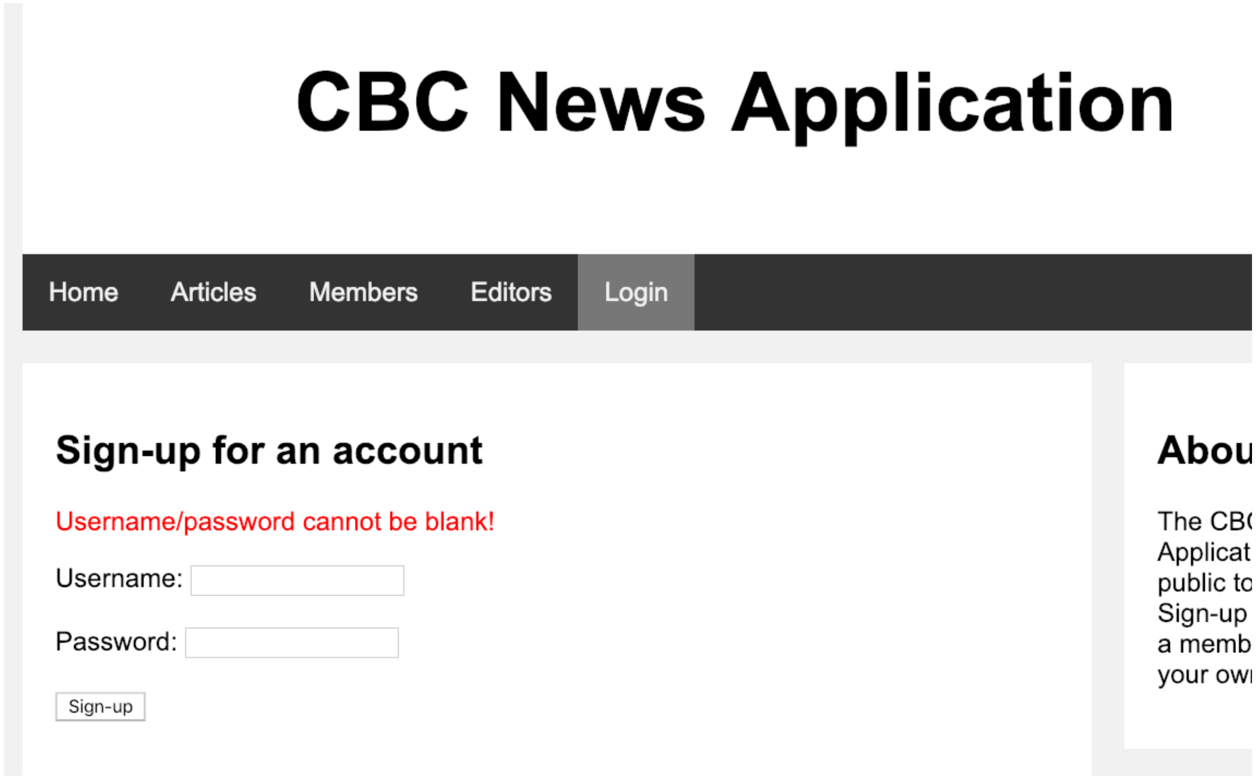
Enter your username and password below to login.

Invalid username and/or password!

The
App
pub
Sig
a m
you

Figure 2 – Login Page Invalid Username/Password

3. **Signup**
    - At the top of the login page just below the “Login” title text, insert a new line of text (“New user? Sign-up for an account!”), where the word Sign-up is a link. When the user clicks on sign-up, they should be brought to a sign-up page that allows them to enter a username and password into text input boxes and click a “Sign-up” button to attempt to create an account. If the username and password entered are not both 1 character or greater in length, the signup page should reload with a red text color error “Username/password cannot be blank!”. If the username and password are both 1 character or greater in length, the new user account should be created in the database with the supplied username and password (with an access level of “member”), and the sign-up page should be re-loaded with a new line of text above the form “**User account created!** Login to access you new account.” (where Login is a link). When the user clicks on Login they should be brought to the Login page, where if they enter the username and password they should be logged into the application. Do not worry about handling the case of a user trying to create an account with an identical username… you can assume nobody will try to do this (though if you want to gracefully handle this case in your application, that’s great too).



Sign-up for an account

Username/password cannot be blank!

Username:

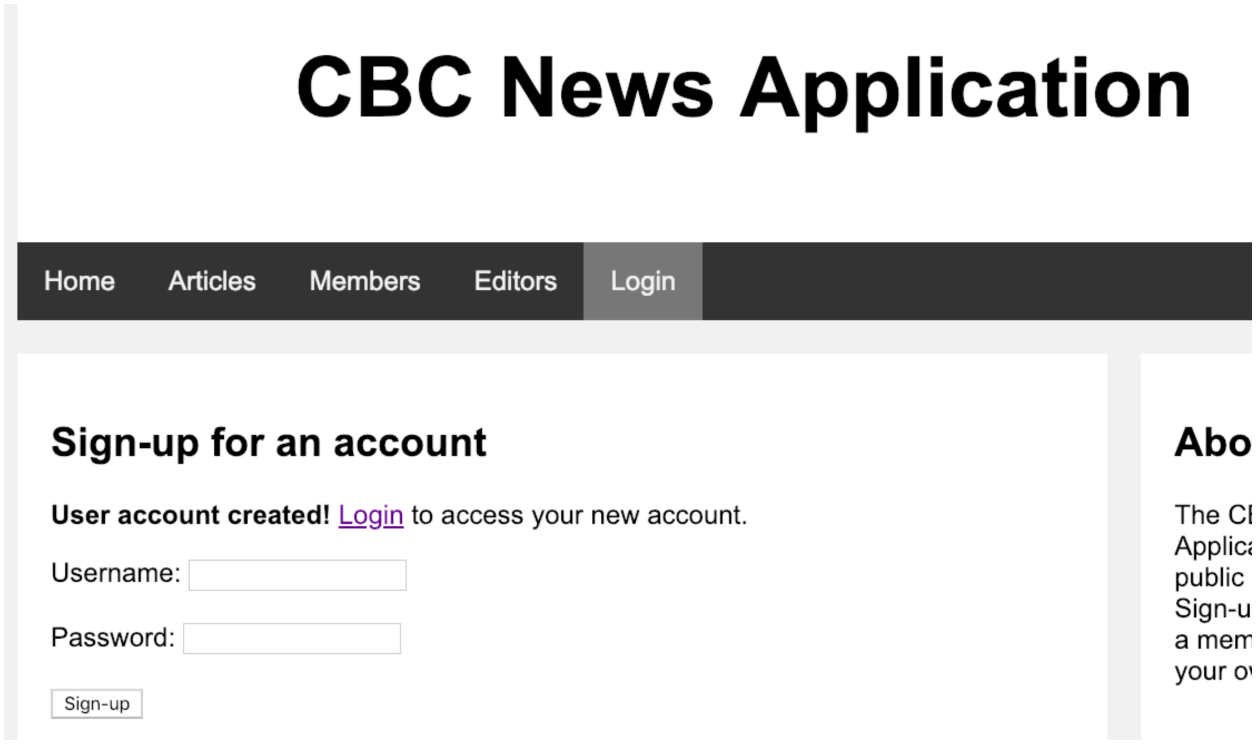
Password:

Sign-up

Abou

The CB(
Applicat
public to
Sign-up
a memb
your ow

Figure 3 - Sign-up Page with Error


CBC News Application

Sign-up for an account

User account created! Login to access your new account.

Username:

Password:

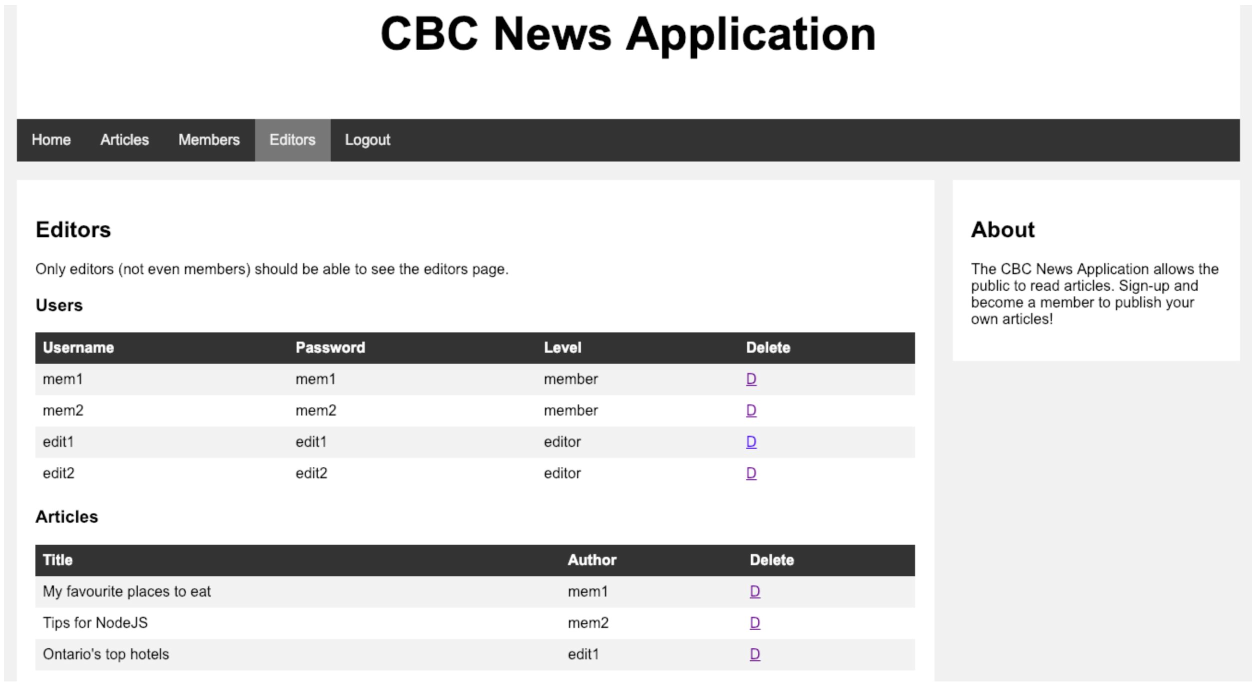
Abo

The CI
Applica
public
Sign-u
a mem
your o

Figure 4 - Sign-up page after account created

4. **Editors Page access**
    - Only a currently logged-in editor-level user should be able to view the Editors page. Public users and logged-in member-level users should not be able to view the Editors page, if they attempt to navigate to the page they should be re-directed to the Home page. Create a middleware to implement this functionality. The middleware should check a session variable that contains the currently logged-in user's access level, and it should use this information to help decide where to send the user.

5. **Editors Page functionality**
    - The Editors page should display a table of all users and a table of all articles. The users table should list usernames, passwords and access levels. The articles table should list article titles and authors (i.e. usernames… but we call them authors on this table, given the context). The articles table should have an additional column with the heading "Delete" and each row in the table associated with an article should have a link with the text "D" in this column. When the user clicks on the "D" in a given row, that article should be deleted from the Articles SQL table. The users table should have an additional column with the heading "Delete" and each row in the table associated with a user should have a link with the text "D" in this column. When the user clicks on the "D" in a given row, that user should be deleted from the Users SQL table, AND any articles authored by that user should be deleted from the Articles SQL table.



Editors

Only editors (not even members) should be able to see the editors page.

Users

Username	Password	Level	Delete
mem1	mem1	member	D
mem2	mem2	member	D
edit1	edit1	editor	D
edit2	edit2	editor	D

Articles

Title	Author	Delete
My favourite places to eat	mem1	D
Tips for NodeJS	mem2	D
Ontario's top hotels	edit1	D

About

The CBC News Application allows the public to read articles. Sign-up and become a member to publish your own articles!

Figure 5 - Editors Page

No other libraries, packages or plugins should be used to implement the above functionality.

Submission

Zip your solution as assignment6.zip and upload it to the dropbox on Avenue.

Marking rubric

Component	Description	Marks
Log middleware	Logs CSV line to file upon each request with relevant data	15
Login	Login based on database table	15
Signup	Signup page functionality	20
Editor page access	Guard access to the editor's page	20
Editor page	Editor page functionality	30
	Total:	/100