CSSD Task 3 Individual Enhancement

Enhanced Login

For my individual enhancement I chose to implement a secure login system. Compared to the basic implementation, this would allow a voter to log into the web application through a login screen and then once logged in would be able to access the ballot screen. Anyone that was not logged in as valid user would not be able to route to certain areas of our directory that they did not have authentication to access; for example if a user wanted to get access to the admin area of the website, or if a user who has already voted wanted to go back to the ballot page and vote for a second time. Other areas I wanted to cover with this enhancement were:

* to log the user out if they did not make a change or move the mouse for the past few minutes
* Only to store passwords in an encrypted form
* Set a limit on the number of times a user could attempt to log in (e.g.4 times) and once this limit was passed their account would be locked and would not be able to get access to it without unlocking it, even with the correct password
* Unlock a locked account by clicking a link in an automated email sent to their registered email address
* Allow users to change their password by them clicking a link on the login page which would send an email to their registered email with a link to a page which would allow them to change their password

Since we moved to using a nodejs server for our middleware there was plenty of node packages that would allow me to easily store secure JSON web tokens as session to allow me to check if they had logged on or not when certain routes are directed to. The package I chose in the end was Passport, due to its ease of use and simple implementation into the middleware of our system.

Once a user attempts to log into the system, multiple checks are done to make sure the voter is valid to login. These consist of checking:

* The inputs form the page are valid
* If the email is connected to a registered user stored on the database
* If their number of login attempts does not exceed 4
* If they have not voted
* If the password they have provided matches the encrypted on stored on the database

If all these test are passed, then the user will be directed to the correct route, else they are routed back the login page with an error message being displayed relevant to why they are an invalid user.

To allow the user to be able to request password changes, an email system had to be created to be able to send the user emails to their registered account with links to reset their password and to unlock their account. For this I used a node package called nodemailer. This allowed me to be able to send users configured emails with links to the reset password route or the unlock account route.

Testing Plan

|  |  |  |
| --- | --- | --- |
| Individual Test Case | Method | Expect Result |
| T3.2  User can unlock their account   * + - T3.3 | 1. Navigate to localhost:3000 2. On the login page, enter the following details and click login:    * Email:    * participant1066@gmail.com    * Password:    * Wrong 3. Repeat this twice more (three times in total), 4. Then enter the correct details:    * Email:    * participant1066@gmail.com    * Password:    * test 5. Log out 6. On login page, enter admin details and login:    * Email:    * participant1066@gmail.com    * Password:    * wrong 7. Click login and see that it fails to log in 8. Go to google.com 9. Sign in with:    * Email:    * [participant1066@gmail.com](mailto:participant1066@gmail.com)    * Password:    * test1066 10. Look at your emails 11. Click on email Titled “EVoting Lockout” 12. Click the link within the email 13. See that it opens on the login page 14. Enter into both new password and confirm password “test” 15. Click reset password 16. See that you are redirected back to the login page 17. On login page, enter admin details:     * Email:     * participant1066@gmail.com     * Password:     * qwerty 18. Click login and see that it fails to log in 19. On login page, enter admin details and login:     * Email:     * participant1066@gmail.com     * Password:     * test 20. See that you log in correctly | The command ‘db.votes.find() should show all votes cast on the system with no information on the voter who cast the vote |
| T.3.3  User can reset password | 1. Navigate to localhost:3000 2. On login page, enter admin details and login:    * Email:    * participant1066@gmail.com    * Password:    * qwerty 3. See that user logs in ok 4. Log out 5. On login page, enter admin details and login:    * Email:    * participant1066@gmail.com    * Password:    * test 6. Click login and see that it fails to log in 7. Enter “participant1066@gmail.com” into email field and click reset password 8. Go to google.com 9. Sign in with:    * Email:    * [participant1066@gmail.com](mailto:participant1066@gmail.com)    * Password:    * test1066 10. Look at your emails 11. Click on email Titled “EVoting Password Reset” 12. Click the link within the email 13. See that the password reset page is opened 14. Enter into both new password and confirm password “test” 15. Click reset password 16. See that you are redirected back to the login page 17. On login page, enter admin details:     * Email:     * participant1066@gmail.com     * Password:     * qwerty 18. Click login and see that it fails to log in 19. On login page, enter admin details and login:     * Email:     * participant1066@gmail.com     * Password:     * test 20. See that you log in correctly | The User will be able to log in with the new password and be rejected if they enter the old password after changing it. |

**UI Testing with Selenium**

With Selenium I created some UI test to make sure validation was working for the login page. The tests I did were:

* Try and log in with only email
* Try and log in with only password
* Try and log in without any inputs
* Try to reset password without using an email

These are included in the Testing folder of the repo.

**API Testing with Postman**

With Postman I created some tests to make sure some of the API calls were working correctly. The tests I did were:

* Request Password Reset With Valid Email
* Request Password Reset With Invalid Email
* Admin Login
* Voter Login
* Try to show password reset page without providing a id

**Charts to show election data**

For the results page I wanted to be able to show the results of an ongoing election with the use of charts. For this I used chart js.

**Chart Result Testing**

|  |  |  |
| --- | --- | --- |
| Test Case | Method | Expected Result |
| Adding a vote will increment the chart data in admin/results= | 1. Navigate to localhost:3000 2. On login page, enter admin details and login:    * Email: [admin@email.com](mailto:admin@email.com)    * Password: qwerty 3. Click view under the results heading 4. Make note of the number of votes shown on the chart 5. Click back, and then click logout 6. Enter details and click login:    * Email: [participant1@email.com](mailto:particpant@email.com)    * Password: qwerty 7. Click the first checkbox on the page, and click submit 8. On a successful vote, click logout 9. On login page, enter admin details and login:    * Email: [admin@email.com](mailto:admin@email.com)    * Password: qwerty 10. Click view under the results heading 11. See that the chart has been incremented by one vote for the party voted for | Chart bar will increase by one, based on whatever party has been voted on. |
| Changing the colour of a party will change the colour for the representing bar on the results chart | 1. Navigate to localhost:3000 2. On login page, enter admin details and login:  * Email: [admin@email.com](mailto:admin@email.com) * Password: qwerty  1. Click view under the results heading 2. Make note of the colour of the “Party 1” bar 3. Click back, and then click “View/Edit” for a party 4. Change the party colour for PArty 1 to Green 5. Go back to the results page and see that the colour for “Party 1” has been changed to Green | Bar chart bar for “Party 1” will have changed colour to green |