Lab 9

System Generated Emails

# Overview

In this exercise you will be increasing the security on a password-based session system, implementing an email-based password reset system.

# Preliminary Work

The first step is to get the code working. You will need a database called ‘nothing’ in MongoDB with two collections:

* user. Upload the data from users.json but feel free to add additional users if you want.
* session. There is no data supplied but the session data will contain: key, expiry, data. The expiry attribute should have a TTL index that expires after 300 seconds to clean up old session data.

You can log in to the account using the credentials as shown in the users.json file. Please make sure to log into both users and look at how the avatar images are being handled.

# Convert Passwords

We learned in the lecture that plain-text passwords are not ideal. Convert the plain text passwords into SHA-256 passwords. Use any online site that you can find to generate the SHA-256 passwords and replace the passwords in the database. You can use Mongo Compass to update the passwords.

Modify the code to use the new hashed passwords.

Verify that you can still log into both accounts with the passwords.

# Fake SMTP Server

Download and run the ‘FakeSMTP’ Server. A zip file is available on D2L that will include the JRE. Unzip the file and run the .exe. Do not try to open the zip file and run the exe without extracting.

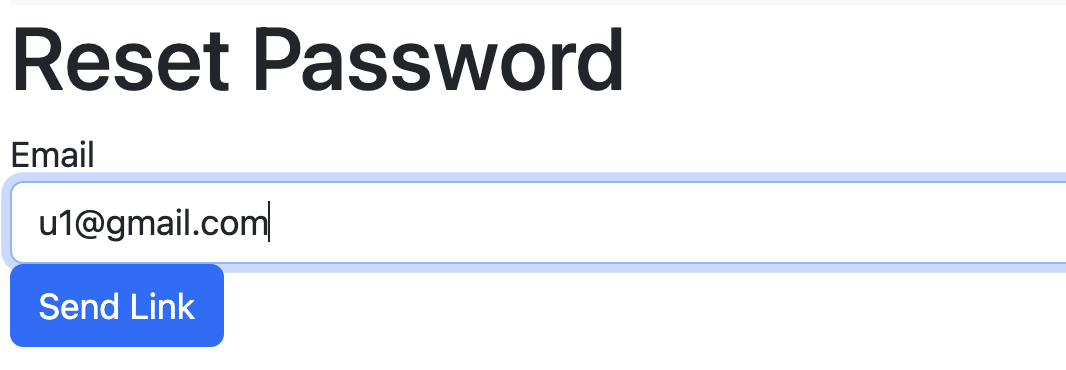
If you are working on a Mac, you can search online for a copy but you will also need to get the correct JRE to run.

# Get Emails to Work

Write a small NodeJS application to send a test message. You can use the example on the PowerPoint slides to try. When you run the program, you should see the message in the FakeSMTP server.

# Password Reset

Put a ‘forgot password’ link on your login page. The link will bring you to a page with a form where the user can enter their email address. See the next section for details of what to do when the button is clicked. The style is not important, please make the functionality works!



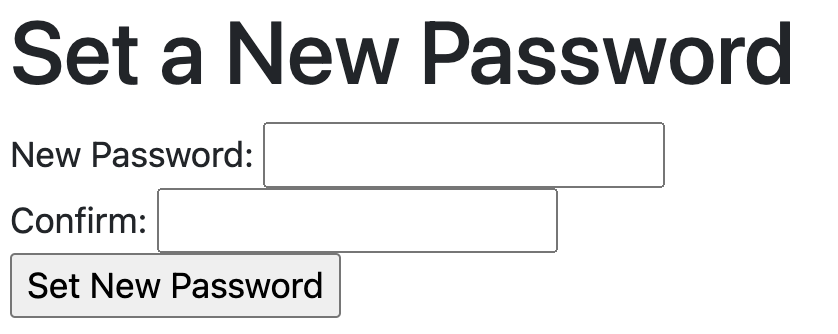
## Handling Password Reset Requests

1. Generate a random key (use a uuid) and add this to the user document in the collection as a field called ‘resetkey’.
2. Add the code required to send an email to the user which includes a link. The reset link would be something like this “http://127.0.0.1/reset-password/232-asdflk-2342”.
3. Display a message to the user to check their email account for the reset link. Even if the user didn’t exist in the collection tell them anyways! This is a security precaution because it means that a potential hacker cannot figure out which email addresses are legitimate.

# Handling the Reset Link

Create a new route (say /reset-password) that does the following:

1. Check that the supplied key is in the user collection. Show an error message and stop if the key does not exist.
2. Give the user a form where they can enter the new password and a confirmation. You need to include the reset key as a hidden field.
3. When the user clicks on ‘Change Password’ then do the next part.



# Resetting Password

The final part of the reset is the setting of the new password. The process is:

1. Check that the password and the confirmation match.
2. Get the user associated with the reset key.
3. Change the password to the new value (don’t forget to use SHA-256).
4. Remove the reset key.
5. Send the user to the login page. The user should be able to log in with the newly set password.