GUIDE B05

Create Endpoints for Deleting The User Profile and Logging Out

Arranged By
Omar Al-Maktary

INFORMATICS ENGINEERING STUDY PROGRAM
INFORMATION TECHNOLOGY DEPARTMENT
STATE POLYTECHNIC OF MALANG
2023

Contents

Create Endpoints for Deleting The User Profile and Logging Out

Objectives

1. Students can create a DELETE endpoint to delete user's data from the database.
2. Students can create two buttons to delete and log out users in the web interface of the application.

Requirements

Having the correct hardware and software components is essential for ensuring the successful execution of the tasks outlined in this guide. The hardware configuration and software required for completing this guide tasks are as the following:

Hardware Specifications

The minimum hardware specifications for running a NodeJS API application on the Windows operating system and using software such as Postman and Visual Studio Code are the following:

Minimum Requirements

- Processor: Intel Core i3 or equivalent.
- RAM: 4 GB.
- Storage: 500 GB HDD with at least 20 GB of available storage.
- Graphics: Integrated graphics card.
- Connectivity: Ethernet and Wi-Fi capabilities.

Recommended Requirements

- Processor: Intel Core i5 or equivalent.
- RAM: 8 GB or more.
- Storage: 256 GB SSD with at least 20 GB of available storage.
- Graphics: Integrated graphics card.
- Connectivity: Ethernet and Wi-Fi capabilities.

Software required

It is important to have the correct software installed on your system to ensure that the application runs smoothly and meets performance expectations. The software required is as follows:

- Operating System: Windows 10 or later.
- NodeJS: Latest stable version installed.
- Visual Studio Code: Latest stable version installed.
- Postman: Latest stable version is installed.

NPM Packages

- nodemon: Automatically restarts Node application on file changes.
- cross-env: Sets environment variables in a cross-platform way.
- jest: Creates and executes tests.
- jest-expect-message: Enhances Jest assertions with custom error messages.
- jest-image-snapshot: Adds image snapshot testing to Jest.
- puppeteer: Node library to control a headless Chrome or Chromium browser.
- supertest: Makes HTTP queries to the application and checks results.
- dotenv: Simplifies management of environment variables.
- express: NodeJS framework for creating apps with routing and middleware.
- ejs: Embedded JavaScript templating.
- express-ejs-layouts: Layout support for EJS in Express.
- mongoose: MongoDB object modeling library for NodeJS.
- bcryptjs: NodeJS library for hashing passwords with the bcrypt algorithm.
- cookie-parser: Parses cookies attached to the incoming HTTP requests.
- cors: Middleware for enabling Cross-Origin Resource Sharing (CORS) in Express.
- express-unless: Defining exceptions to other middleware functions in Express.
- jsonwebtoken: Manage authentication by creating and verifying tokens.

Resource

- Documents: Guide B05
- Tests: api/testB05.test.js, web/testB05 .test.js

Task Description

Students understand how to create an endpoint for deleting a user’s profile data. This endpoint will return the user data by using an access token. Students should also understand how to create a web interface that deletes the user’s account.

Start Coding

To create an endpoint for deleting user data, students should understand the following table which outlines the structure and goals of the endpoint. Note that there is an additional header for the authorization which is the token obtained from the login or registration process. This token has the user id encoded which can be used to identify the user.

DELETE “/api/v1/profile” ENDPOINT STRUCTURE			
API Endpoint Path	*Request Method*	*Response Format*	*Description*
“/api/v1/profile”	DELETE	JSON	Delete the user from the database
Additional Headers			
Key	*Value*		*Description*
“Authorization”	“Bearer accessToken”		AccessToken is the value of the token obtained from the login process
Request Parameters (no parameters needed)			
Response Parameters			
Parameter	*Type*		*Description*
“message”	String		A message indicating the status of the response.
Success Responses			
HTTP Status Code	*Response*		
200	{ "message": " User Deleted Successfully" }		
Error Responses			
HTTP Status Code	*Response*		
401	{message “Unauthorized”}		
400	{message: “User Delete Failed”}		
500	{ error object }		

Table 1 DELETE “/api/v1/profile” ENDPOINT STRUCTURE

Follow the steps below to complete the code for this guide document:

1. In the “auth.service.js” file, copy and complete the following code.

```
async function deleteProfile(id) {
 // Find the user and delete it by using FindByIdAndDelete method
 const user = // Write your code here;
 // if the user is not found throw an error
 // The error message should be "User Delete Failed"
 if (!user) // Write your code here;
 // If the user is found return the message "User Deleted Successfully"
}
```
Figure 1 Delete Profile Data Function Code

2. Export the "deleteProfile" function at the end of the "auth.service.js" file.
3. In the "controllers/api/auth.controller.js" file, copy and complete the following code.

```
function deleteProfile(req, res, next) {
 // Get the user id from the req.user.id
 // Call the authServices.deleteProfile(id)
 authServices
   .deleteProfile(id)
   .then((results) => // If successful, return a 200 status code and the results)
   .catch((err) => // If unsuccessful, call the next() function with the error);
}
```
Figure 2 Delete Profile Controller Function Code

4. Export the "deleteProfile" function at the end of the "controllers/api/auth.controller.js" file.
5. In the "routes/api/auth.routes.js" file, import the deleteProfile function from the API controller.
6. Finally, Create a new DELETE route with the "/profile" path.

Running The API Application

For this guide and development purposes the command "npm run dev" is used to execute the command "nodemon server.js" which will run the "server.js" using the nodemon package. This package allows the server to reload if any changes occur in the code of the application.

Run the development command "npm run dev" in the terminal and notice the console message.

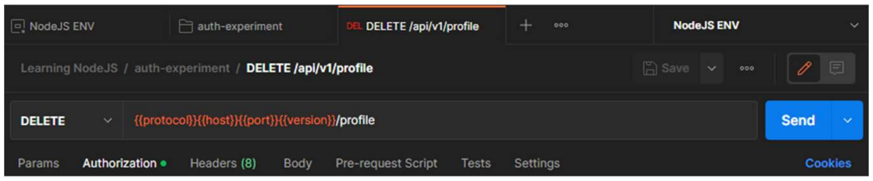
Testing The API Application

In this section, several tests in different ways will be explored to verify the results of the student's work on this document.

Using Postman

To test results from this guide on Postman, follow these steps:

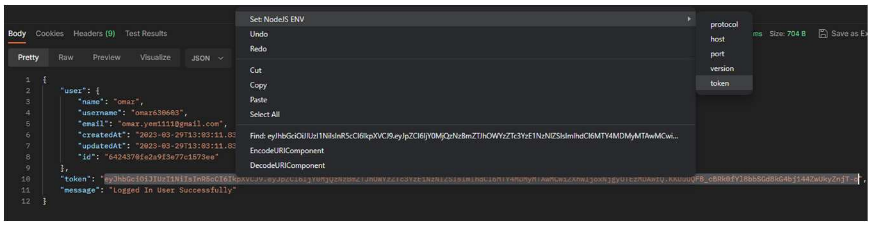
1. In the “auth-experiment” collection, create a DELETE request with the name “GET /api/v1/profile”.
2. Make sure that the environment created is being used by selecting it from the top right option and then fill in the URL in the DELETE request as the following: “{{protocol}}{{host}}{{port}}{{version}}/profile”



```
DELETE  {{protocol}}{{host}}{{port}}{{version}}/profile
```

Figure 3 Postman Request Configuration Bar

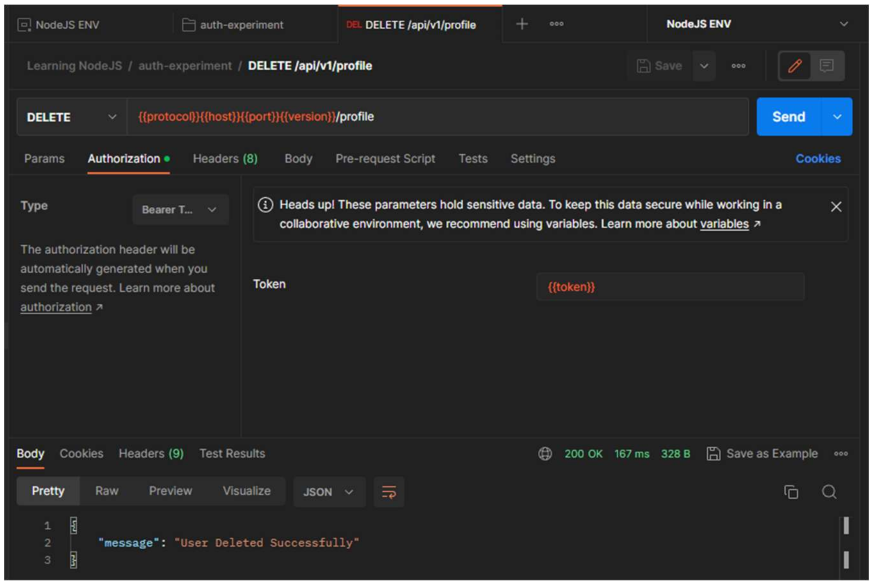
3. If the token is still active then skip to step 6, and check the token by requesting the endpoint to fetch the user’s profile. If the response returns the user’s data then the token is still active
4. In the Authorization tab, choose the “Bearer Token” as the type of authorization and fill the token field with “{{token}}” which should be created in the environment variables.
5. Request the login route created in the previous guide and save the token returned into the environment variable as the following:



```
{
    "user": {
        "name": "omar",
        "username": "omar630603",
        "email": "omar.yem1111@gmail.com",
        "createdAt": "2023-03-29T13:03:11.83
        "updatedAt": "2023-03-29T13:03:11.83
        "id": "6424370fe2a9f3e77c1573ee"
    },
    "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6Ikp[illegible]",
    "message": "Logged In User Successfully"
}
```

Figure 4 Postman Saving Token

6. Go back to the DELETE profile request and click “Send” then wait for the response. Postman should show results as the following if everything is working correctly:



The authorization header will be automatically generated when you send the request. Learn more about authorization ↗

Heads up! These parameters hold sensitive data. To keep this data secure while working in a collaborative environment, we recommend using variables. Learn more about variables ↗

Token {{token}}

```
{
    "message": "User Deleted Successfully"
}
```

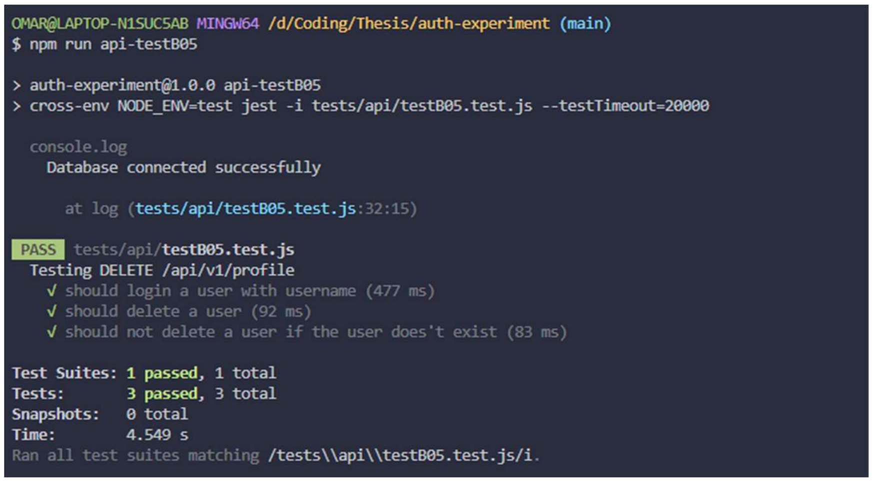
Figure 5 Postman Request Results

Running The API Test File

Note: Sometimes the test will have an error of time limit, try to re-run the test or increase the testTimeout in scripts of the “package.json” file.

Verify results by following these steps:

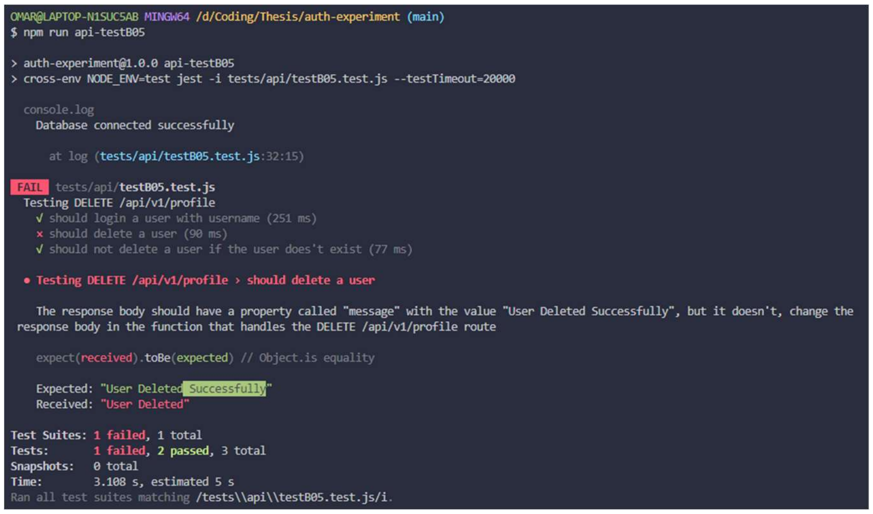
1. Copy the file "testB05.test.js" from the "api" folder within the "tests" folder for this material to the "tests/api" folder of your project base directory.
1. Run the fill in the VSCode integrated terminal by running this command “npm run api-testB05” and then wait for results.
2. If everything is correct and working well the results in the terminal should look as the following:



```
OMAR@LAPTOP-N1SUC5AB MINGW64 /d/Coding/Thesis/auth-experiment (main)
$ npm run api-testB05

> auth-experiment@1.0.0 api-testB05
> cross-env NODE_ENV=test jest -i tests/api/testB05.test.js --testTimeout=20000

  console.log
    Database connected successfully

      at log (tests/api/testB05.test.js:32:15)

 PASS  tests/api/testB05.test.js
  Testing DELETE /api/v1/profile
    √ should login a user with username (477 ms)
    √ should delete a user (92 ms)
    √ should not delete a user if the user does't exist (83 ms)

Test Suites: 1 passed, 1 total
Tests:       3 passed, 3 total
Snapshots:   0 total
Time:        4.549 s
Ran all test suites matching /tests\\api\\testB05.test.js/i.
```

Figure 6 Successful Test Results

3. If the test failed and it shows an error similar to the following figure, the error shows feedback for the cause of the error:



```
OMAR@LAPTOP-N1SUC5AB MINGW64 /d/Coding/Thesis/auth-experiment (main)
$ npm run api-testB05

> auth-experiment@1.0.0 api-testB05
> cross-env NODE_ENV=test jest -i tests/api/testB05.test.js --testTimeout=20000

  console.log
    Database connected successfully

      at log (tests/api/testB05.test.js:32:15)

 FAIL  tests/api/testB05.test.js
  Testing DELETE /api/v1/profile
    √ should login a user with username (251 ms)
    × should delete a user (90 ms)
    √ should not delete a user if the user does't exist (77 ms)

  ● Testing DELETE /api/v1/profile › should delete a user

    The response body should have a property called "message" with the value "User Deleted Successfully", but it doesn't, change the
 response body in the function that handles the DELETE /api/v1/profile route

    expect(received).toBe(expected) // Object.is equality

    Expected: "User Deleted Successfully"
    Received: "User Deleted"

Test Suites: 1 failed, 1 total
Tests:       1 failed, 2 passed, 3 total
Snapshots:   0 total
Time:        3.108 s, estimated 5 s
Ran all test suites matching /tests\\api\\testB05.test.js/i.
```

Figure 7 Failed Test Results

Try to find out why the test failed and fix it until the test result shows successful results.

Creating The Web Interface

In this section, the web interface for the profile page will be created. The same basic endpoint explained in the previous sections will be implemented in a web page that can show the user's data.

To start working on the web interface, follow these steps:

1. In the "controllers/web/auth.controller.js" file, copy and complete the following code:

```
function deleteProfile(req, res, next) {
 // Get the user id from the req.user.id
 // Call the authServices.deleteProfile(id)
 authServices.deleteProfile(id).then((results) => {
   // Clear the cookie
   // Redirect to the home page
   res.clearCookie("token");
 });
}
function logout(req, res, next) {
 // Clear the cookie
 // Redirect to the home page
 return res.redirect("/");
}
```
Figure 8 "controllers/web/auth.controller.js" Delete Profile and Logout Functions Code

2. Export the "deleteProfile and logout" function at the end of the "controllers/web/auth.controller.js" file.
3. In the "routes/web/auth.routes.js" file, import the "deleteProfile and logout" function from the web controller.
4. Add this new route as the following.

```
router.post("/profile/delete", isLoggedIn, deleteProfile);
router.get("/logout", isLoggedIn, logout);
```
Figure 9 "routes/web/auth.routes.js" New Routes

5. No need to create new pages because the logout button and delete form has been created on the profile page.

   .

Running and Testing The Web Interface

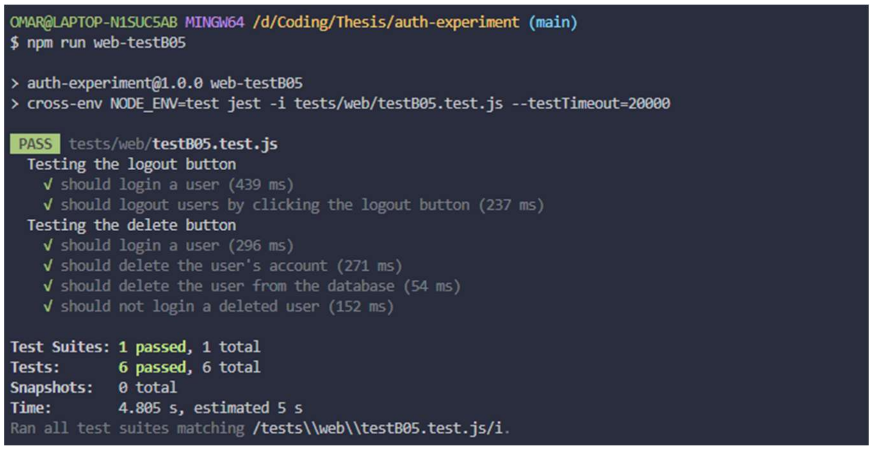
If the application still running from the previous exercise then try to visit the following link http://localhost:8080/. If the application is not running in the terminal then use the command "npm run dev" to start the app.

If the user is not logged in then log in using the login page and it will automatically redirect to the profile page if the code is working correctly.

Upon visiting the URL, click on the profile nav bar link. Click on the log-out button or the delete button and the website should be redirected to the home page.

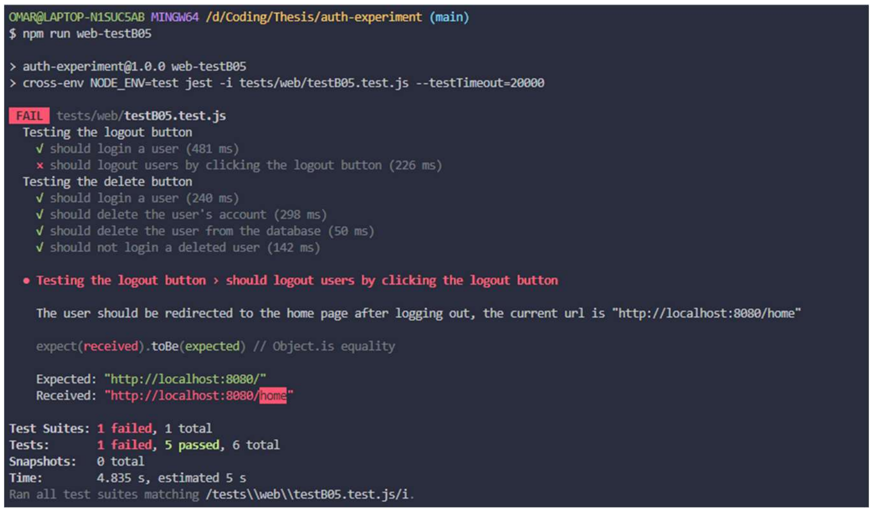
Copy the file "testB05.test.js" and paste it to the folder "/tests/web" in your project directory. After that, run the command "npm run web-testB05" and notice the results.

If everything is correct and working well the results in the terminal should look as the following:



```
OMAR@LAPTOP-N1SUC5AB MINGW64 /d/Coding/Thesis/auth-experiment (main)
$ npm run web-testB05

> auth-experiment@1.0.0 web-testB05
> cross-env NODE_ENV=test jest -i tests/web/testB05.test.js --testTimeout=20000

 PASS  tests/web/testB05.test.js
  Testing the logout button
    √ should login a user (439 ms)
    √ should logout users by clicking the logout button (237 ms)
  Testing the delete button
    √ should login a user (296 ms)
    √ should delete the user's account (271 ms)
    √ should delete the user from the database (54 ms)
    √ should not login a deleted user (152 ms)

Test Suites: 1 passed, 1 total
Tests:       6 passed, 6 total
Snapshots:   0 total
Time:        4.805 s, estimated 5 s
Ran all test suites matching /tests\\web\\testB05.test.js/i.
```

Figure 10 Successful Web Test Results

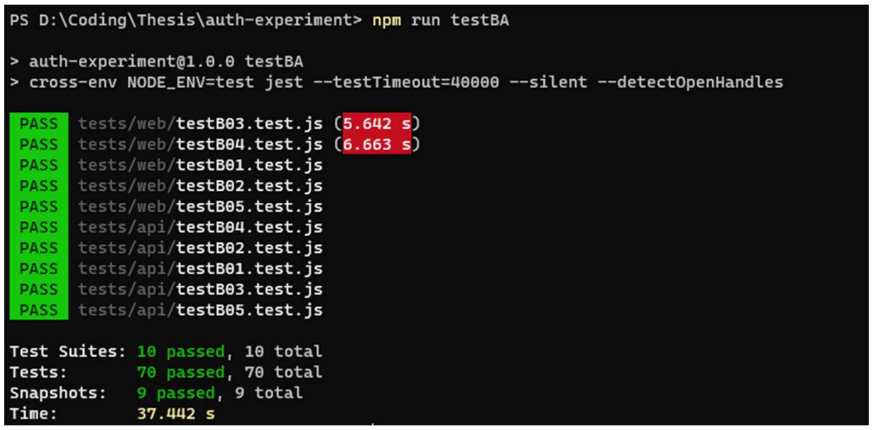


```
OMAR@LAPTOP-N1SUC5AB MINGW64 /d/Coding/Thesis/auth-experiment (main)
$ npm run web-testB05

> auth-experiment@1.0.0 web-testB05
> cross-env NODE_ENV=test jest -i tests/web/testB05.test.js --testTimeout=20000

 FAIL  tests/web/testB05.test.js
  Testing the logout button
    √ should login a user (481 ms)
    × should logout users by clicking the logout button (226 ms)
  Testing the delete button
    √ should login a user (240 ms)
    √ should delete the user's account (298 ms)
    √ should delete the user from the database (50 ms)
    √ should not login a deleted user (142 ms)

  ● Testing the logout button › should logout users by clicking the logout button

    The user should be redirected to the home page after logging out, the current url is "http://localhost:8080/home"

    expect(received).toBe(expected) // Object.is equality

    Expected: "http://localhost:8080/"
    Received: "http://localhost:8080/home"

Test Suites: 1 failed, 1 total
Tests:       1 failed, 5 passed, 6 total
Snapshots:   0 total
Time:        4.835 s, estimated 5 s
Ran all test suites matching /tests\\web\\testB05.test.js/i.
```

Figure 11 Failed Web Test Results

If you face a similar error try to figure out the reason for the problem until the test shows successful results.

FInall step is to run all the tests from the start of this learning material until this material. To run all tests run the command “npm run testBA” which will run all the tests in the “tests” folder.



```
PS D:\Coding\Thesis\auth-experiment> npm run testBA

> auth-experiment@1.0.0 testBA
> cross-env NODE_ENV=test jest --testTimeout=40000 --silent --detectOpenHandles

 PASS  tests/web/testB03.test.js (5.642 s)
 PASS  tests/web/testB04.test.js (6.663 s)
 PASS  tests/web/testB01.test.js
 PASS  tests/web/testB02.test.js
 PASS  tests/web/testB05.test.js
 PASS  tests/api/testB04.test.js
 PASS  tests/api/testB02.test.js
 PASS  tests/api/testB01.test.js
 PASS  tests/api/testB03.test.js
 PASS  tests/api/testB05.test.js

Test Suites: 10 passed, 10 total
Tests:       70 passed, 70 total
Snapshots:   9 passed, 9 total
Time:        37.442 s
```

Figure 12 Successfully Ran All The Tests

Results

This document outlines the intended outcomes of the fifth meeting for the second material on the topic of web programming using NodeJS. Students should learn how to create an API endpoint for deleting a user's account from the database. The students should also learn how to create the functions needed for handling the logout and delete requests on the web.