**CCT College Dublin**

**Assessment Cover Page**

|  |  |
| --- | --- |
| **Module Title:** | Server-side Programming |
| **Assessment Title:** | Final Project - 60% |
| **Lecturer Name:** | **Kayoum Khbuli** |
| **Student Full Name:** | Luis Ramirez, Telmuun dunia, Alejo Santos |
| **Student Number:** | 2023169, 2023306, 2023197. |
| **Assessment Due Date:** | Sunday, 22 December 2024, 11:59 PM |
| **Date of Submission:** |  |

**Declaration**

|  |
| --- |
| By submitting this assessment, I confirm that I have read the CCT policy on Academic Misconduct and understand the implications of submitting work that is not my own or does not appropriately reference material taken from a third party or other source. I declare it to be my own work and that all material from third parties has been appropriately referenced. I further confirm that this work has not previously been submitted for assessment by myself or someone else in CCT College Dublin or any other higher education institution. |

Contents

[A) Data Validation and Database Insertion: 1](#_Toc185597300)

[• Implement JavaScript methods to validate the submitted CSV data, this is provided in the index.js file. 1](#_Toc185597301)

[• If validation fails for a record, throw an error message indicating the index position of the failed record. 1](#_Toc185597302)

[• Insert only the valid records into the database, make sure the table name is “mysql\_table”. 3](#_Toc185597303)

[• Ensure the database table column names are camel cased as follows i.e “first\_name”. 3](#_Toc185597304)

[B) Form and Data Validation Rules. 5](#_Toc185597305)

[Creating a connection with the database. 7](#_Toc185597306)

[Creating a request from the server. 8](#_Toc185597307)

[Sending data from the server to the database. 9](#_Toc185597308)

[Creating a request from the server. 9](#_Toc185597309)

[Sending data from the server to the database. 10](#_Toc185597310)

[D) Secure Connection Protocols and XSS Mitigation: 11](#_Toc185597311)

[HTTPS Implementation 11](#_Toc185597312)

[12](#_Toc185597313)

[Content Security Policy (CSP) Headers 14](#_Toc185597314)

[User Input Sanitization 15](#_Toc185597315)

[Regular Software Updates and Security Patches 15](#_Toc185597316)

[References 15](#_Toc185597317)

[GeeksforGeeks. “How to Create HTTPS Server with Node.js ?” *GeeksforGeeks*, 29 Sept. 2021, www.geeksforgeeks.org/how-to-create-https-server-with-node-js/. Accessed 18 Dec. 2024. 16](#_Toc185597318)

[Victory, Nwani. “How to Implement SSL/TLS Pinning in Node.js.” *Snyk*, 29 Aug. 2023, snyk.io/blog/ssl-tls-pinning-node-js/. Accessed 18 Dec. 2024. 16](#_Toc185597319)

[William, Stallings. *Cryptography and Network Security - Principles and Practice, 7th Edition*. global edition ed., Pearson Education India, 2017. 16](#_Toc185597320)

[Book. 16](#_Toc185597321)

# A) Data Validation and Database Insertion:

• Implement JavaScript methods to validate the submitted CSV data, this is provided in the index.js file.

A screen shot of a computer

Description automatically generated

Creating a connection to our database with the relevant host address, username and password.

A screen shot of a computer code

Description automatically generated

Regex validation that is being used to check the data input from the sample, and according to the validation the decision to add the information to the database or not will be decided.

• If validation fails for a record, throw an error message indicating the index position of the failed record.

A screen shot of a computer code

Description automatically generated

Making a function that checks the data to our regex, split the name and throw an error if it does not not contain first and last name.

A screenshot of a computer code

Description automatically generated

The regex validation being used to check the input from the sample data.

A screen shot of a computer

Description automatically generated

Returning verified data, of a single row.

A computer screen with text on it

Description automatically generated

Creating a new function that will run through the sample data, dividing them between commas ‘,’ and return the value after validating using checkData function.

A black background with white text

Description automatically generated

• Insert only the valid records into the database, make sure the table name is “mysql\_table”.

A screen shot of a computer code

Description automatically generated

Insert the data we checked and validated to our MySQL database by utilizing INSERT INTO query and print a message to user indicating how many users were added from the sample data.

• Ensure the database table column names are camel cased as follows i.e “first\_name”.

A black screen with yellow text

Description automatically generated

And lastly, actually running the functions we declared above with our sample data.

A screenshot of a computer

Description automatically generated

We added a console.log command to show how the data is being stored and validated through our function. This was mostly used for troubleshooting while working with the file.

  
Above is our final result. The user Michael Johnson was not added to the database as they had an invalid input of “Null” on the age field, which means our data insertion is working perfectly as intended.

A screenshot of a computer

Description automatically generated

Our sample data was successfully inserted to our database, with an auto incrementing primary key “id” as an extra column for future functionality. And our database follows the required camel cased name structure.

# B) Form and Data Validation Rules.

A screenshot of a computer

Description automatically generatedWe created a form webpage to collect customers' information so they can be contacted for information about cars. The structure of this webpage contains five attributes that we are going to collect and post into the database, those are first name, last name, email, phone, age, and Eircode.

We use JavaScript to enforce the validation of all the attributes.

A screen shot of a computer code

Description automatically generatedFirst and last names must be only alphanumeric and no

A computer screen with text

Description automatically generatedEmail must contain the @ symbol and the email structure with “.com” for example.

Phone number must have a length of 10 digits.

A screenshot of a computer program

Description automatically generated

Age must be over 18 and under 70 to be accepted.

A screen shot of a computer program

Description automatically generated

A screen shot of a computer code

Description automatically generatedThe Eircode must start with a number, be alphanumeric, and have a length of 6 characters.

If one of these inputs fails in the validation, a windows alert will pop on the screen indicating the error.

Creating a connection with the database.

**A screen shot of a computer

Description automatically generated**Using the MySQL package installed, we can create the credentials to log in the database using the connection variable.

A computer screen shot of a program

Description automatically generatedOnce the credentials are done, we made a query to create a new table if not exist using the MySQL language, with all the attributes that we are going to use after. This will save us time and prevent us from misspelling something wrong when creating a table manually in MySQL Workbench.

A screenshot of a computer program

Description automatically generated

Creating a request from the server.

Using the .get() method we can request data from the server without modifying it. In this case, we use the form.html as a response to an HTTP request.

A screenshot of a computer program

Description automatically generated

Sending data from the server to the database.

With the post.() method we can send data from the user input in the webpage to the database. In this case, we have to specify the endpoint, the “/submit” in the form.html file and give a structure to the data coming from the form file.

A screen shot of a computer

Description automatically generated

Creating a request from the server.

Using the .get() method we can request data from the server without modifying it. In this case, we use the form.html as a response to an HTTP request.

A screenshot of a computer program

Description automatically generated

Sending data from the server to the database.

With the post.() method we can send data from the user input in the webpage to the database. In this case, we have to specify the endpoint, the “/submit” in the form.html file and give a structure to the data coming from the form file.

A screen shot of a computer

Description automatically generated

# D) Secure Connection Protocols and XSS Mitigation:

HTTPS Implementation

In this first part, we will discuss how HTTPS (Hypertext Transfer Protocol Secure) implementation can protect data leaks and thwart attacks.

Hypertext Transfer Protocol Secure better known as HTTPS ensures that all communication between the client and server is encrypted using SSL/TLS protocols. This implement secure communication between a web browser and a web server

SSL/TLS is a security mechanism that helps to protect against man-in-the-middle (MITM) and other attacks. It does this by ensuring that a client, as a Node.js app, only connects with a pre-verified digital certificate. So creating an HTTPS server in Node.js ensures secure communication between your server and clients. Using HTTPS you are adding a layer of security essential for handling sensitive information.

To create HTTPS the first step would be creating the SSL (Secure Sockets Layer) certificate.

Doing this is pretty straightforward as open git bash on your laptop running the command

“openssl req -nodes -new -x509 -keyout server.key -out server.cert”

So after this, you will get two files one is the server.cert and the other one is server.key.

The next step is creating a form to send a message to the server through a post request.

# A screen shot of a computer program Description automatically generated

To initialize the project in the terminal the command “npm init”



After we ran this command, we had to install the packages and libraries in our project

A screenshot of a computer program

Description automatically generated

A screenshot of a computer program

Description automatically generated

So we created an HTTPS server using createServer function. We passed the certificate and key files of the SSL certificates as options objects in createrServer function. We manage to get and post requests using express in Node.js.

A screen shot of a computer program

Description automatically generated

We ran this from the root directory of the project and then opened the browser typing localhost3000 and your webpage running with HTTPS is there.

This is the way HTTPS works and also how to implement to create a HTTPS server using Node.js

Content Security Policy (CSP) Headers

This is a feature that helps to prevent or minimize the risk of certain types of security threats.

The primary use case for CSP is to control which resources. In particular JavaScript resources, a document is allowed to load. This is mainly used as a defense against cross-site scripting (XXS) attacks.

XXS is a type of security vulnerability in web applications where an attacker injects malicious scripts into web pages viewed by other users. One way to avoid this is the CSP.

This could be an example of a script tag that links a malicious source:

A screen shot of a computer

Description automatically generated

Also, it could be an inline event as an image or a javascript URL, etc. So, in order to provide protection against all of these we need a CSP. A CSP could disable all these attacks.

The main purpose of the CSP is to keep you safe from these attacks but also has another purpose including defending against clickjacking and helping to ensure that a site’s pages will be loaded over HTTPS.

User Input Sanitization

This refers to the process of cleaning or “sanitizing” user inputs to ensure safety. It doesn’t affect [input devices](https://www.ninjaone.com/blog/what-is-an-input-device/) but instead involves removing or modifying data that could potentially [lead to system vulnerabilities](https://www.ninjaone.com/blog/what-is-vulnerability-management/) or errors

There are a few methods to apply this, the most common are:

The blacklist and whitelist. The first one creates a list of known malicious inputs and then it proceeds to block them all.

The second one creates a list of approved inputs and only allowing those to pass through. It is considered a more secure approach as only known safe inputs are permitted.

This brings a several benefits as prevention of inclusion and injection attacks, compliance with industry regulations. Etc.

Regular Software Updates and Security Patches

This keeps the software, frameworks, libraries, and operating systems up to date. With this we make sure that vulnerabilities, bugs and wearknesses are fixed patch by patch to keep the system from exploitation by attackers.

The best way to implement this maintaining an updating schedule, setting a routine, week or monthly to check for updates to your software and libraries. For Node.js

You have a few commands as:



Checking for outdates packages.

A black background with white text

Description automatically generated

Update all packages to their latest compatible versions

So in conclusion regular updates and security patches are essential to maintaining a secure and stable application environment. By keeping software up to date you are reducing the attack surface, preventing data breaches and ensuring the applications keep working properly.

# References

GeeksforGeeks. “How to Create HTTPS Server with Node.js ?” *GeeksforGeeks*, 29 Sept. 2021, www.geeksforgeeks.org/how-to-create-https-server-with-node-js/. Accessed 18 Dec. 2024.

Victory, Nwani. “How to Implement SSL/TLS Pinning in Node.js.” *Snyk*, 29 Aug. 2023, snyk.io/blog/ssl-tls-pinning-node-js/. Accessed 18 Dec. 2024.

William, Stallings. *Cryptography and Network Security - Principles and Practice, 7th Edition*. global edition ed., Pearson Education India, 2017.

Book.