**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**

[**Introduction.** 3](#_Toc185603037)

[**Data validation and database insertion.** 1](#_Toc185603038)

[Validating the CSV data in the index file. 1](#_Toc185603039)

[Inserting only valid records into the database from the CSV variable. 3](#_Toc185603040)

[**Form and Data Validation Rules.** 5](#_Toc185603041)

[Creating a connection with the database. 7](#_Toc185603042)

[Creating a request from the server. 8](#_Toc185603043)

[Sending data from the server to the database. 9](#_Toc185603044)

[Creating a request from the server. 9](#_Toc185603045)

[Sending data from the server to the database. 10](#_Toc185603046)

[**Middleware for Request Handling.** 10](#_Toc185603047)

[**Secure Connection Protocols and XSS Mitigation.** 11](#_Toc185603048)

[HTTPS Implementation. 11](#_Toc185603049)

[Content Security Policy (CSP) Headers. 13](#_Toc185603050)

[User Input Sanitization. 14](#_Toc185603051)

[Regular Software Updates and Security Patches. 14](#_Toc185603052)

[**Conclusion.** 15](#_Toc185603053)

[**References** 16](#_Toc185603054)

# **Introduction.**

# **Data validation and database insertion.**

A computer screen shot of a computer code

Description automatically generatedValidating the CSV data in the index file.

To create a connection with the database we have to register the credentials to log in. Using the MySQL package and the method createConnection we can log in using the user, password, host and database names attributes.

A screen shot of a computer code

Description automatically generated

Regex validation 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.

Error-handling.

A screen shot of a computer code

Description automatically generated

A screenshot of a computer code

Description automatically generatedMaking a function that checks the data to our regex, splits the name and throws an error if it does not contain the first and last name. The name validation split the data using the “ , ” coma, and with some restrictions as if, fullName length is under 2 (but not equal) or (||) remove the spaces behind and after the name from the first name and second name. All inputs are validated and checked with the error handle if it does not match with the regex variable.

One by one, all the input are being tested.

A screen shot of a computer

Description automatically generated

If all the data is correct, this will return the verified data printed on screen.

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

Inserting only valid records into the database from the CSV variable.

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 the user indicating how many users were added from the sample data.

A black screen with yellow text

Description automatically generated

Lastly, run 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 into our database, with an auto-incrementing primary key “id” as an extra column for future functionality of our database following the required camel-case structure.

# **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 window 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 into 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

# **Middleware for Request Handling.**

A computer screen shot of a program code

Description automatically generatedWe implemented a middleware that connects with the database and checks if the schema already exists and if the format is the right one for the incoming input.

This other middleware just listen the port 3000 and send a message to the user with the connection status and port.

A screenshot of a computer screen

Description automatically generated

This middleware checks the connection to the database and returns an status message.

A black screen with white text

Description automatically generated

# **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 implements secure communication between a web browser and a web server. (Victory, 2023)

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. (GeeksforGeeks, 2021)

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.

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

A screen shot of a computer program

Description automatically generated

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

A screenshot of a computer program

Description automatically generatedA 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 defence 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 allows those to pass through. It is considered a more secure approach as only known safe inputs are permitted.
* This brings several benefits as the prevention of inclusion and injection attacks and 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 weaknesses are fixed patch by patch to keep the system from exploitation by attackers. (Stallings, 2017)

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:

A black background with white text

Description automatically generatedChecking for outdates packages.

Update all packages to their latest compatible versions

# **Conclusion.**

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. 2021.** GeeksforGeeks. *GeeksforGeeks.* [Online] 11 29, 2021. www.geeksforgeeks.org/how-to-create-https-server-with-node-js/.

**Stallings, William. 2017.** *Cryptography and Network Security - Principles and Practice.* India : global edition ed., 2017.

**Victory, Nwani. 2023.** SNYK. *snyk.io.* [Online] 10 29, 2023. snyk.io/blog/ssl-tls-pinning-node-js/.