JSON Practice

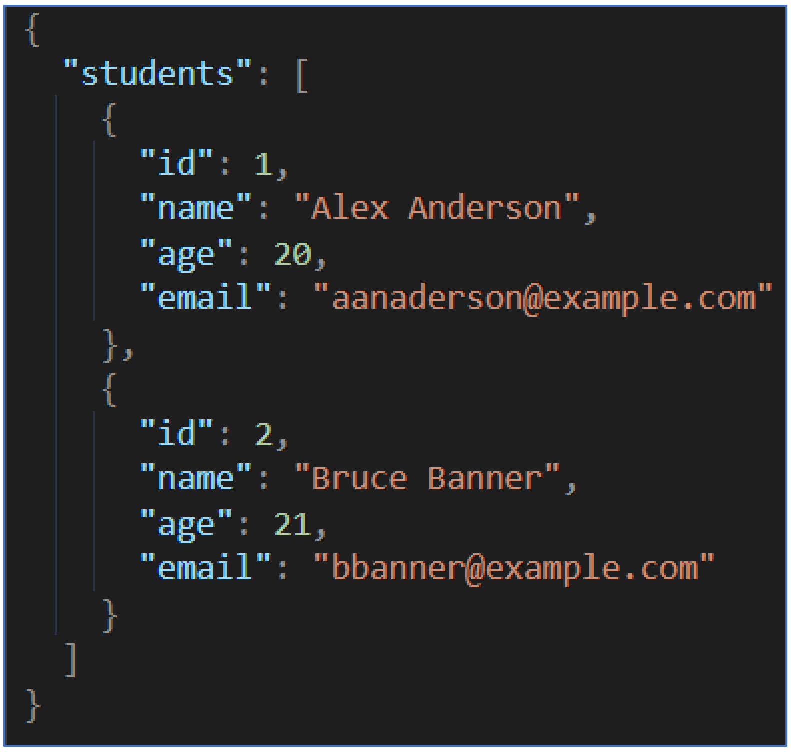
Lab Objectives:

In this lab, you will focus on manipulating JSON files using PHP. A number of exercises will be provided to you so that you understand and utilise relevant PHP functions to read, modify, write and delete JSON content. The integration of PHP and JSON will allow you to explore dynamic web application development possibilities. This lab will therefore provide you with valuable skills and prepare you for real-world scenarios involving JSON data manipulation and processing.

Instructions:

1. Open any text editor software, e.g. Notepad, Atom, Visual Studio, etc.

2. Create an JSON file called "students.json" which holds student information similar to the below.



```json
{
  "students": [
    {
      "id": 1,
      "name": "Alex Anderson",
      "age": 20,
      "email": "aanaderson@example.com"
    },
    {
      "id": 2,
      "name": "Bruce Banner",
      "age": 21,
      "email": "bbanner@example.com"
    }
  ]
}
```

3. Make sure that you modify the above code to include at least 5 student records in the “students.json” file.

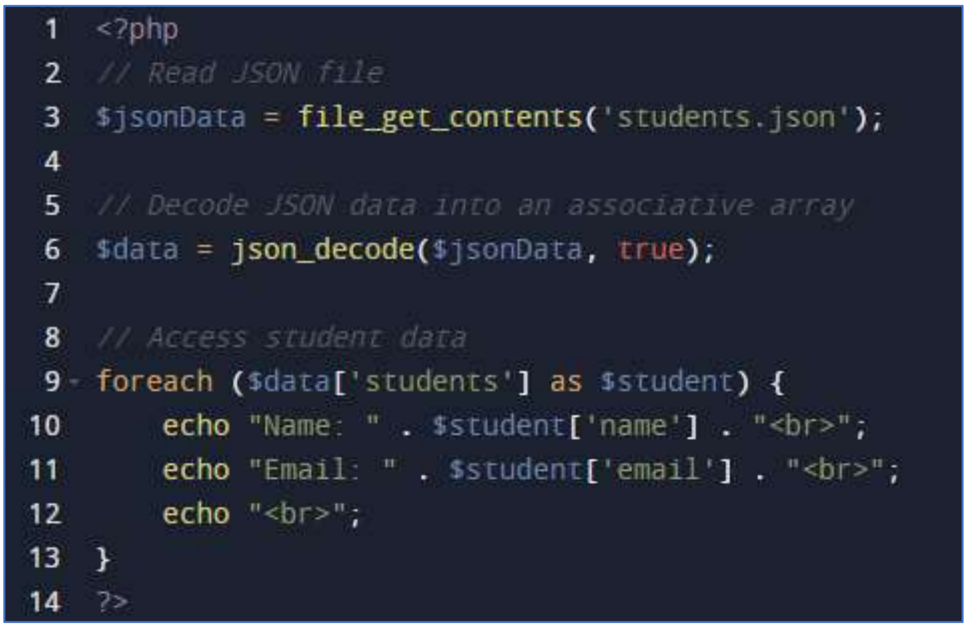
4. Using PHP, you will now perform the following operations.

a. Read and Display JSON Data

-> Read the contents of the "students.json" file using PHP.

-> For this, you should create a new file called "JSON_Read.php".

Sample Code:



```php
<?php
// Read JSON file
$jsonData = file_get_contents('students.json');

// Decode JSON data into an associative array
$data = json_decode($jsonData, true);

// Access student data
foreach ($data['students'] as $student) {
    echo "Name: " . $student['name'] . "<br>";
    echo "Email: " . $student['email'] . "<br>";
    echo "<br>";
}
?>
```

Code Explanation:

$jsonData = file_get_contents('students.json');

This line reads the contents of the 'students.json' file and assigns it to the variable **$jsonData**. The function **file_get_contents()** is used to read the file and retrieve its contents as a string.

$data = json_decode($jsonData, true);

This line decodes the JSON data stored in $jsonData using the **json_decode()** function and converts it into a PHP associative array.

The second argument "true" indicates that we want the JSON data to be returned as an associative array rather than as an object.

foreach ($data['students'] as $student) { … }

This line initiates a loop that iterates over each element within the **'students'** array in the **$data** associative array. Each element represents one student's data.

echo "Name: " . $student['name'] . "<br>";

This line retrieves and displays the name of each student. The **$student** variable holds the current element in the loop which is an associative array representing a student's data. The 'name' key is used to access the name value for the current student and it is concatenated with the string "Name:" using the **"."** operator. The **echo** statement displays this concatenated string. The <br> tag adds a line break for better readability.

echo "Email: " . $student['email'] . "<br>";

This line retrieves and displays the email address of each student in a similar manner to the above step.

Modify the sample code to display the student's id and age information as well.

b. Add a New Student to the JSON File

-> Add a new student record to the JSON file with the following details.

ID: 6

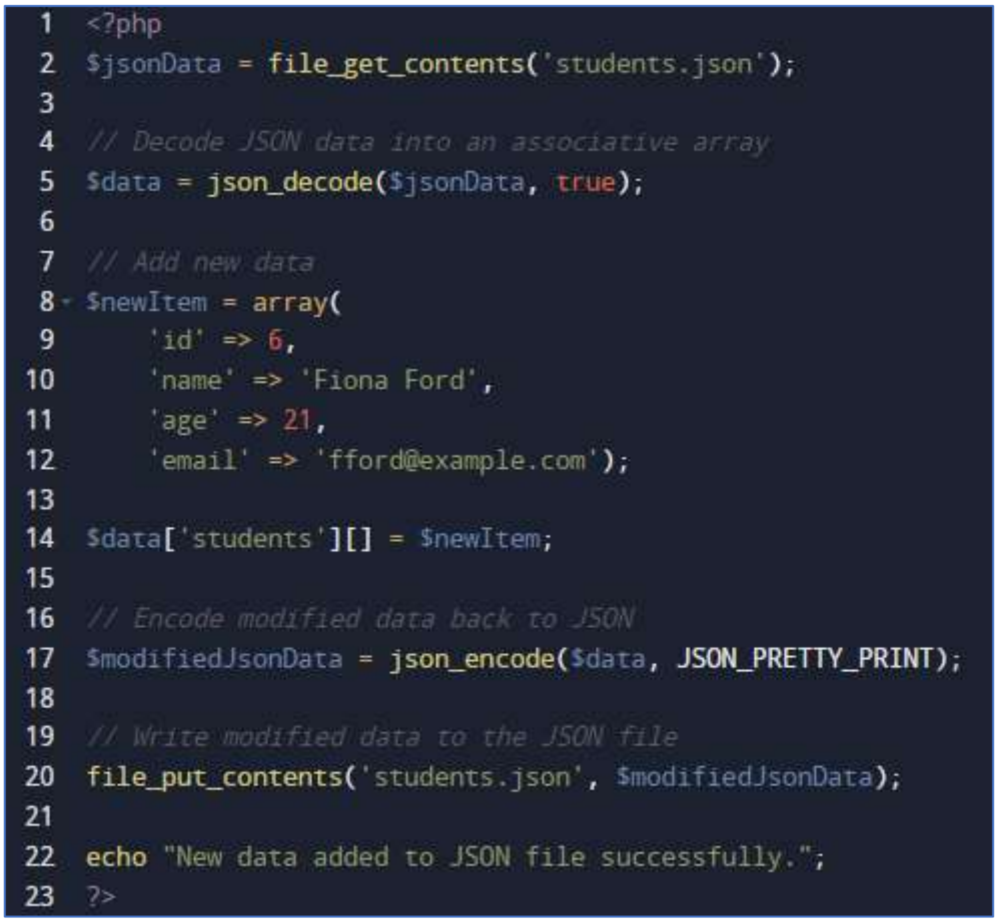
Name: Fiona Ford

Age: 21

Email: fford@example.com

-> For this, you should create a new file called "JSON_Add.php".

Sample Code:



```php
<?php
$jsonData = file_get_contents('students.json');

// Decode JSON data into an associative array
$data = json_decode($jsonData, true);

// Add new data
$newItem = array(
    'id' => 6,
    'name' => 'Fiona Ford',
    'age' => 21,
    'email' => 'fford@example.com');

$data['students'][] = $newItem;

// Encode modified data back to JSON
$modifiedJsonData = json_encode($data, JSON_PRETTY_PRINT);

// Write modified data to the JSON file
file_put_contents('students.json', $modifiedJsonData);

echo "New data added to JSON file successfully.";
?>
```

Code Explanation:

$jsonData = file_get_contents('students.json');

$data = json_decode($jsonData, true);

See previous example for explanation on these lines of code.

$newItem = array(...);

This line defines a new array called **$newItem** that contains the data for the new student to be added. The array elements correspond to the keys and values of the student's attributes, that is, 'id', 'name', 'age' and 'email'.

$data['students'][] = $newItem;

This line appends the **$newItem** array to the **'students'** array within the $data associative array. In other words, it adds the new student data to the existing collection of students.

$modifiedJsonData = json_encode($data, JSON_PRETTY_PRINT);

This line encodes the modified **$data** array back into JSON format using the **json_encode()** function. The **JSON_PRETTY_PRINT** flag is used to format the JSON output with indentation and line breaks for better readability.

file_put_contents('students.json', $modifiedJsonData);

This line writes the modified JSON data stored in **$modifiedJsonData** back to the 'students.json' file. The **file_put_contents()** function is used to write the contents to the file, overwriting the previous data.

echo "New data added to JSON file successfully.";

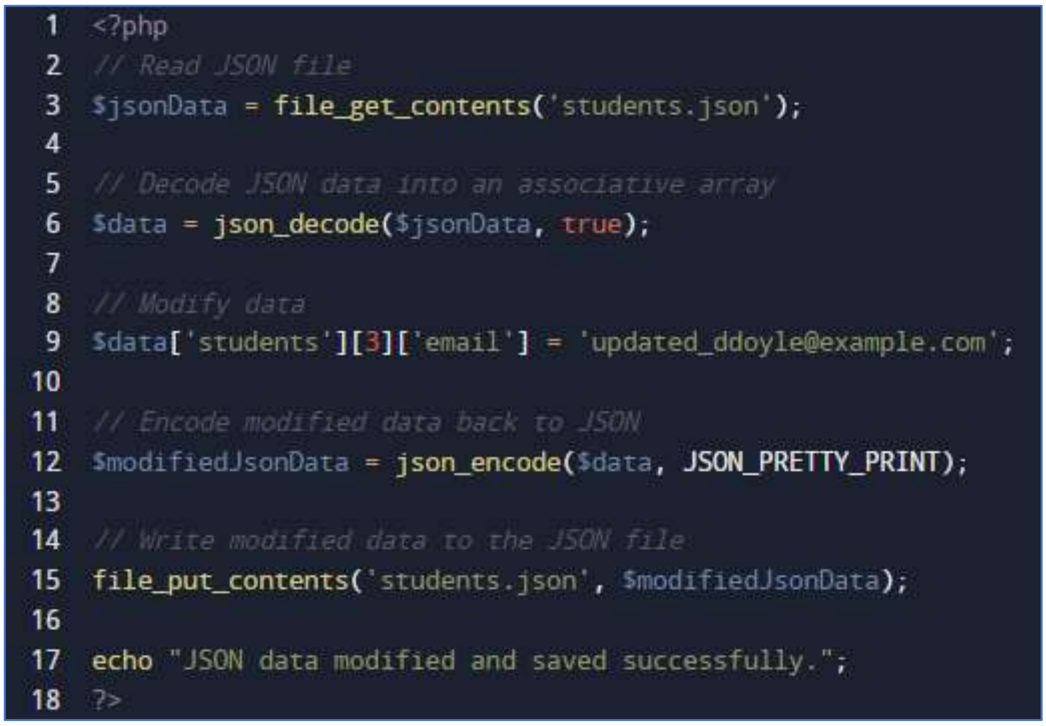
This line displays a success message indicating that the new data has been added to the JSON file.

c. Update Student Data in the JSON File

-> Find the fourth student (ID 4) and update their email to "**updated_ddoyle@example.com**" in the JSON file.

-> For this, you should create a new file called “JSON_Update.php”.

Sample Code:



```php
<?php
// Read JSON file
$jsonData = file_get_contents('students.json');

// Decode JSON data into an associative array
$data = json_decode($jsonData, true);

// Modify data
$data['students'][3]['email'] = 'updated_ddoyle@example.com';

// Encode modified data back to JSON
$modifiedJsonData = json_encode($data, JSON_PRETTY_PRINT);

// Write modified data to the JSON file
file_put_contents('students.json', $modifiedJsonData);

echo "JSON data modified and saved successfully.";
?>
```

Code Explanation:

$jsonData = file_get_contents('students.json');

$data = json_decode($jsonData, true);

See previous examples for explanation on these lines of code.

$data['students'][3]['email'] = 'updated_ddoyle@example.com';

This line modifies the email address of the student at **index 3 (fourth student)** in the 'students' array within the **$data** associative array. It assigns the new email address **'updated_ddoyle@example.com'** to the 'email' key of that student.

$modifiedJsonData = json_encode($data, JSON_PRETTY_PRINT);

file_put_contents('students.json', $modifiedJsonData);

echo "JSON data modified and saved successfully.";

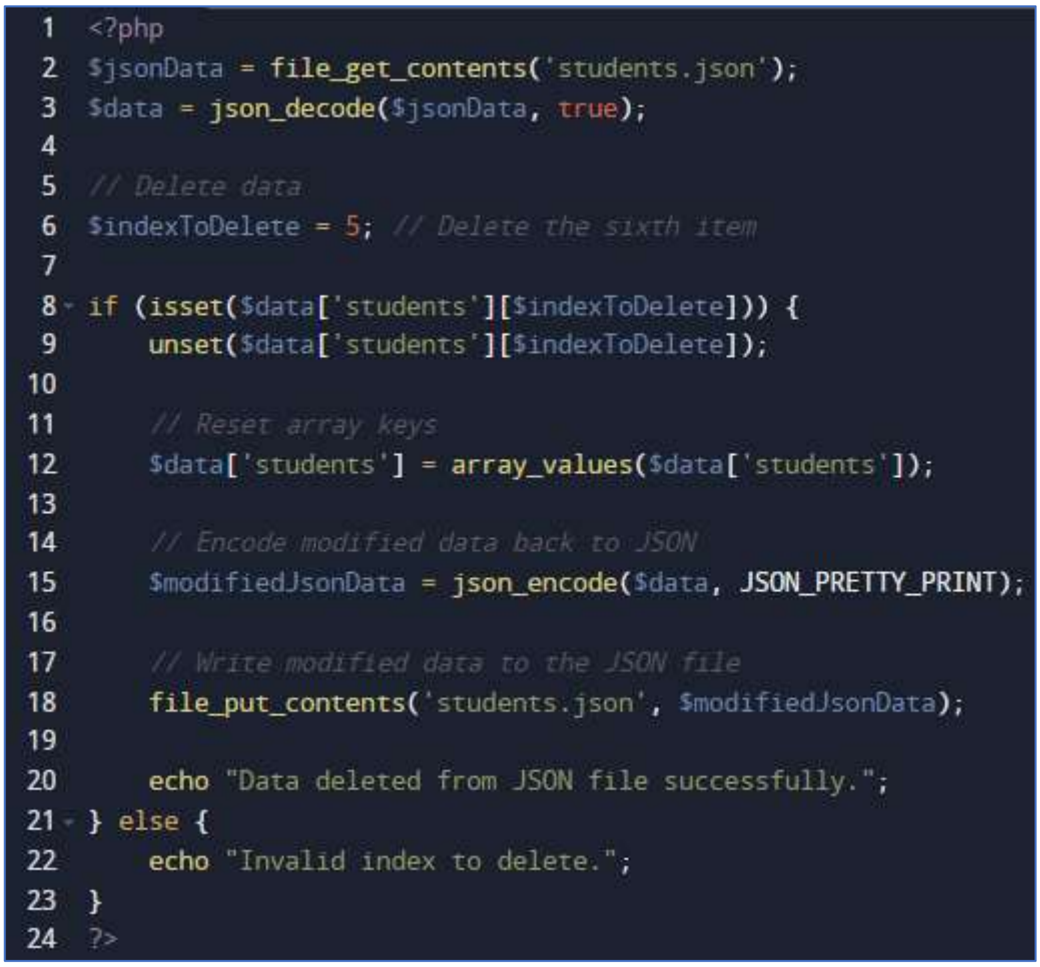
See previous examples for explanation on these lines of code.

d. Delete a Student in the JSON File

-> Remove the student with **ID 6** from the JSON file.

-> For this, you should create a new file called "JSON_Delete.php".

Sample Code:



```php
<?php
$jsonData = file_get_contents('students.json');
$data = json_decode($jsonData, true);

// Delete data
$indexToDelete = 5; // Delete the sixth item

if (isset($data['students'][$indexToDelete])) {
    unset($data['students'][$indexToDelete]);

    // Reset array keys
    $data['students'] = array_values($data['students']);

    // Encode modified data back to JSON
    $modifiedJsonData = json_encode($data, JSON_PRETTY_PRINT);

    // Write modified data to the JSON file
    file_put_contents('students.json', $modifiedJsonData);

    echo "Data deleted from JSON file successfully.";
} else {
    echo "Invalid index to delete.";
}
?>
```

Code Explanation:

```
$jsonData = file_get_contents('students.json');
$data = json_decode($jsonData, true);
```

See previous examples for explanation on these lines of code.

```
$indexToDelete = 5;
```

This line sets the index of the item to be deleted. In this case, the **sixth item** will be deleted since the index is set to 5 (array indices start from 0).

```
if (isset($data['students'][$indexToDelete])) { ... }
```

This condition checks if the specified index to delete exists in the 'students' array within $data.

```
unset($data['students'][$indexToDelete]);
```

This line removes the item at the specified index from the 'students' array using the **unset()** function.

```
$data['students'] = array_values($data['students']);
```

This line resets the array keys of the 'students' array to maintain sequential numeric keys after deleting an item. The **array_values()** function is used to reassign the array keys.

$modifiedJsonData = json_encode($data, JSON_PRETTY_PRINT);

file_put_contents('students.json', $modifiedJsonData);

echo "Data deleted from JSON file successfully.";

See previous examples for explanation on these lines of code.

else { echo "Invalid index to delete."; }

This else block is executed if the specified index to delete does not exist in the 'students' array. It displays an **error message** indicating an invalid index.

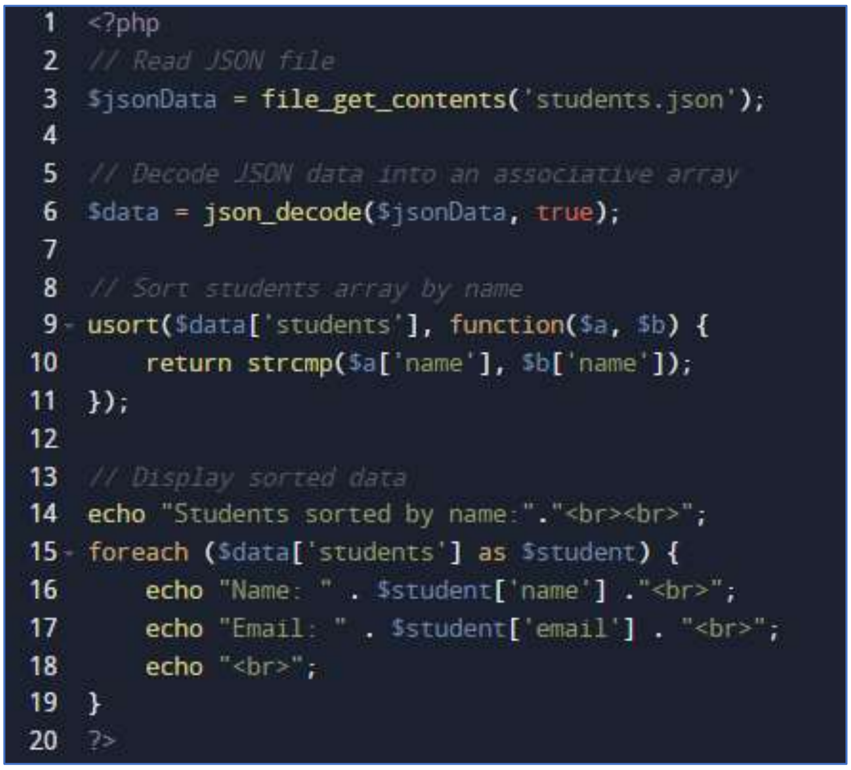
Modify the sample code so that the IDs of the student records remain sequential even when deleting a student which is not at the end of the JSON file.

e. Sorting JSON Data

-> Sort the student records in alphabetical order

-> For this, you should create a new file called "JSON_Sort.php".

Sample Code:



```php
<?php
// Read JSON file
$jsonData = file_get_contents('students.json');

// Decode JSON data into an associative array
$data = json_decode($jsonData, true);

// Sort students array by name
usort($data['students'], function($a, $b) {
    return strcmp($a['name'], $b['name']);
});

// Display sorted data
echo "Students sorted by name:"."<br><br>";
foreach ($data['students'] as $student) {
    echo "Name: " . $student['name'] ."<br>";
    echo "Email: " . $student['email'] . "<br>";
    echo "<br>";
}
?>
```

Code Explanation:

$jsonData = file_get_contents('students.json');

$data = json_decode($jsonData, true);

See previous examples for explanation on these lines of code.

usort($data['students'], function($a, $b) { ... });

This line sorts the 'students' array within the $data associative array using the **usort()** function. The **usort()** function is used to sort an array by values using a user-defined comparison function. In this case, the comparison function compares the 'name' values of two students using **strcmp()** for string comparison.

Note that:

The **strcmp()** function is a built-in PHP function used for string comparison.

It compares two strings and returns an integer value indicating the result of the comparison. See below.

0 if the two strings are equal.

A negative integer if $a is less than $b.

A positive integer if $a is greater than $b.

foreach ($data['students'] as $student) { ... }

This loop iterates over each element in the sorted 'students' array within the $data associative array.

echo "Name: " . $student['name'] ."<br>";

echo "Email: " . $student['email'] . "<br>";

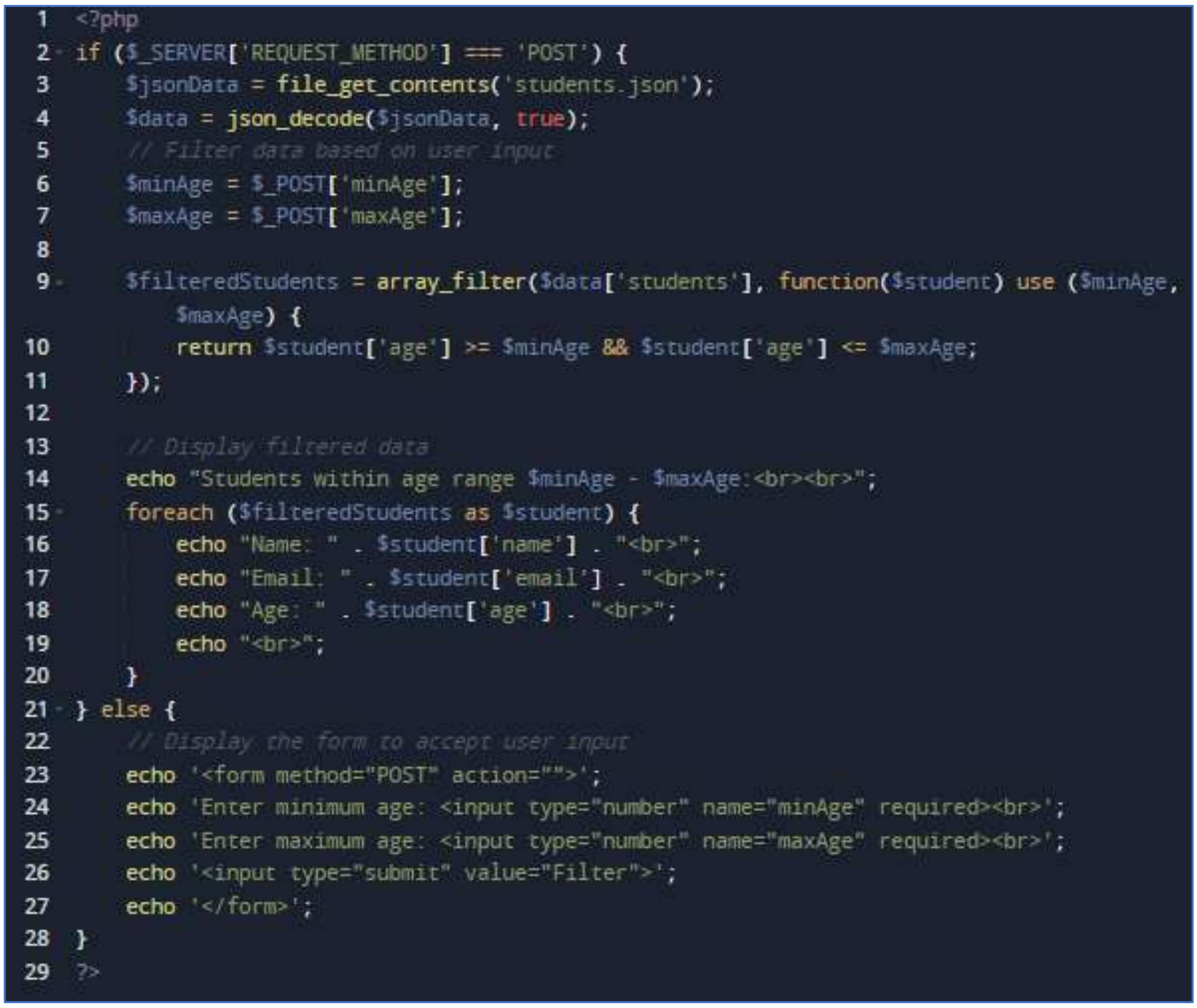
These lines retrieve and display the name and email of each student in the sorted array.

f. Filtering JSON Data

-> Filter students based on a pre-defined range for their age.

-> For this, you should create a new file called "JSON_Filter.php".

Sample Code:



```php
<?php
if ($_SERVER['REQUEST_METHOD'] === 'POST') {
    $jsonData = file_get_contents('students.json');
    $data = json_decode($jsonData, true);
    // Filter data based on user input
    $minAge = $_POST['minAge'];
    $maxAge = $_POST['maxAge'];

    $filteredStudents = array_filter($data['students'], function($student) use ($minAge, $maxAge) {
        return $student['age'] >= $minAge && $student['age'] <= $maxAge;
    });

    // Display filtered data
    echo "Students within age range $minAge - $maxAge:<br><br>";
    foreach ($filteredStudents as $student) {
        echo "Name: " . $student['name'] . "<br>";
        echo "Email: " . $student['email'] . "<br>";
        echo "Age: " . $student['age'] . "<br>";
        echo "<br>";
    }
} else {
    // Display the form to accept user input
    echo '<form method="POST" action="">';
    echo 'Enter minimum age: <input type="number" name="minAge" required><br>';
    echo 'Enter maximum age: <input type="number" name="maxAge" required><br>';
    echo '<input type="submit" value="Filter">';
    echo '</form>';
}
?>
```

Code Explanation:

if ($_SERVER['REQUEST_METHOD'] === 'POST') { ... }

This condition checks if the request method is **POST** indicating that the user has submitted the form.

$jsonData = file_get_contents('students.json');

$data = json_decode($jsonData, true);

See previous examples for explanation on these lines of code.

$minAge = $_POST['minAge']; and $maxAge = $_POST['maxAge'];

These lines retrieve the user-inputted minimum and maximum ages from the submitted form using the **$_POST superglobal** array.

$filteredStudents=array_filter($data['students'], function($student) use ($minAge, $maxAge) { ... });

This line filters the **'students'** array within **the $data** associative array based on the provided age range. It uses the **array_filter()** function along with an anonymous function as the callback. The callback function checks if each student's age falls within the specified range using the **>=** and **<=** comparison operators.

The subsequent code within the **foreach loop** displays the filtered student data, including their name, email, and age.

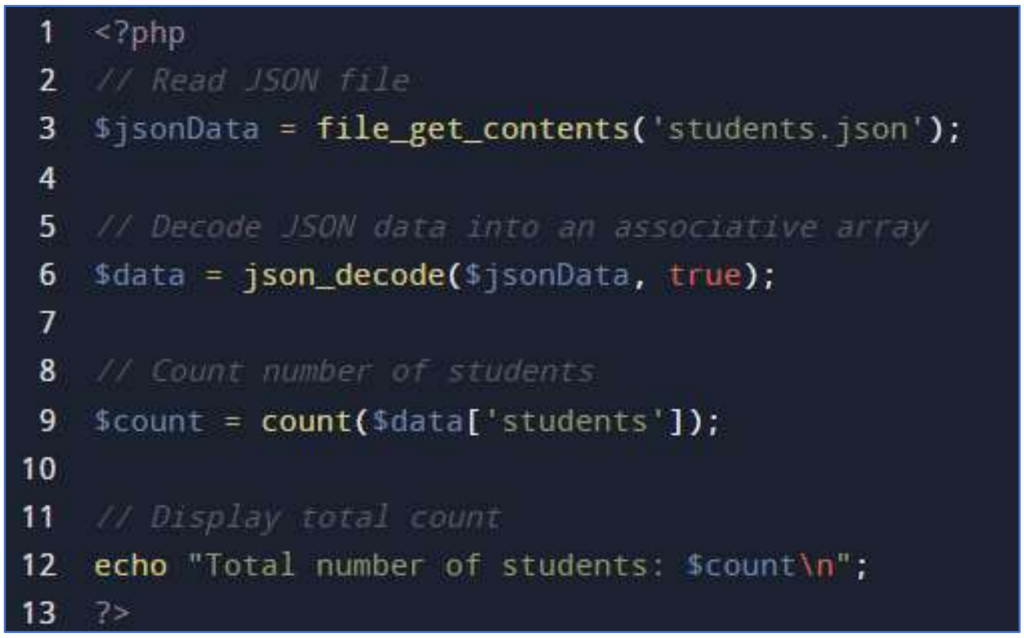
The **else block** is executed when the request method is not POST which means the user has not yet submitted the form. It displays an HTML form that prompts the user to enter the minimum and maximum ages and submit the form for filtering.

g. Counting JSON Data

-> Count the total number of students available in the JSON file.

-> For this, you should create a new file called “JSON_Count.php”.

Sample Code:



```php
<?php
// Read JSON file
$jsonData = file_get_contents('students.json');

// Decode JSON data into an associative array
$data = json_decode($jsonData, true);

// Count number of students
$count = count($data['students']);

// Display total count
echo "Total number of students: $count\n";
?>
```

Code Explanation:

$jsonData = file_get_contents('students.json');

$data = json_decode($jsonData, true);

See previous examples for explanation on these lines of code.

$count = count($data['students']);

This line uses the **count()** function to determine the number of elements in the 'students' array within the $data associative array.

It returns the count of students and assigns it to the variable **$count**.

echo "Total number of students: $count\n";

This line displays the total number of students by concatenating the string "Total number of students: " with the value of **$count**.

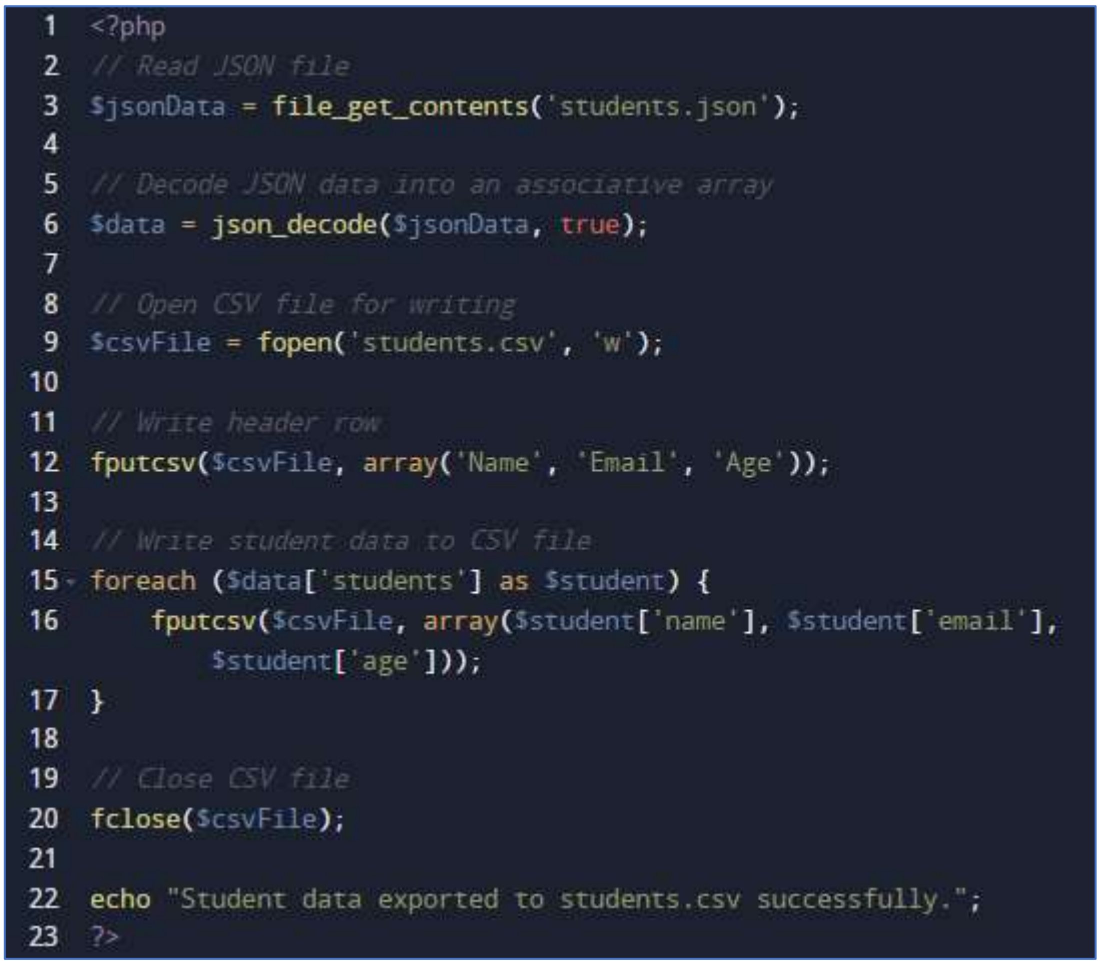
The **\n** is a newline character used for formatting.

h. Exporting JSON Data

-> Export the whole content of the “students.json” file to an external format which can then be opened using Microsoft Excel or similar spreadsheet software.

-> For this, you should create a new file called “JSON_Export.php”.

Sample Code:



```php
<?php
// Read JSON file
$jsonData = file_get_contents('students.json');

// Decode JSON data into an associative array
$data = json_decode($jsonData, true);

// Open CSV file for writing
$csvFile = fopen('students.csv', 'w');

// Write header row
fputcsv($csvFile, array('Name', 'Email', 'Age'));

// Write student data to CSV file
foreach ($data['students'] as $student) {
    fputcsv($csvFile, array($student['name'], $student['email'],
        $student['age']));
}

// Close CSV file
fclose($csvFile);

echo "Student data exported to students.csv successfully.";
?>
```

Code Explanation:

$jsonData = file_get_contents('students.json');

$data = json_decode($jsonData, true);

See previous examples for explanation on these lines of code.

$csvFile = fopen('students.csv', 'w');

This line opens a CSV file named **'students.csv'** in **write mode ('w')** and assigns the file handle to the variable **$csvFile**.

The **fopen()** function is used to open the file.

fputcsv($csvFile, array('Name', 'Email', 'Age'));

This line writes the **header row** to the CSV file. The **fputcsv()** function writes an array of values as a line in the CSV file whilst automatically formatting them as comma-separated values. In this case, the array contains the column headers: 'Name', 'Email', and 'Age'.

foreach ($data['students'] as $student) { ... }

This loop iterates over each element in the 'students' array within $data.

fputcsv($csvFile, array($student['name'], $student['email'], $student['age']))

This line writes the student data to the CSV file. The **fputcsv()** function is used again to write an array of values as a line in the CSV file.

The array contains the name, email, and age values extracted from the current student's associative array.

fclose($csvFile);

This line closes the CSV file using the file handle stored in **$csvFile**.

The **fclose()** function is used to close the file.

echo "Student data exported to students.csv successfully.";

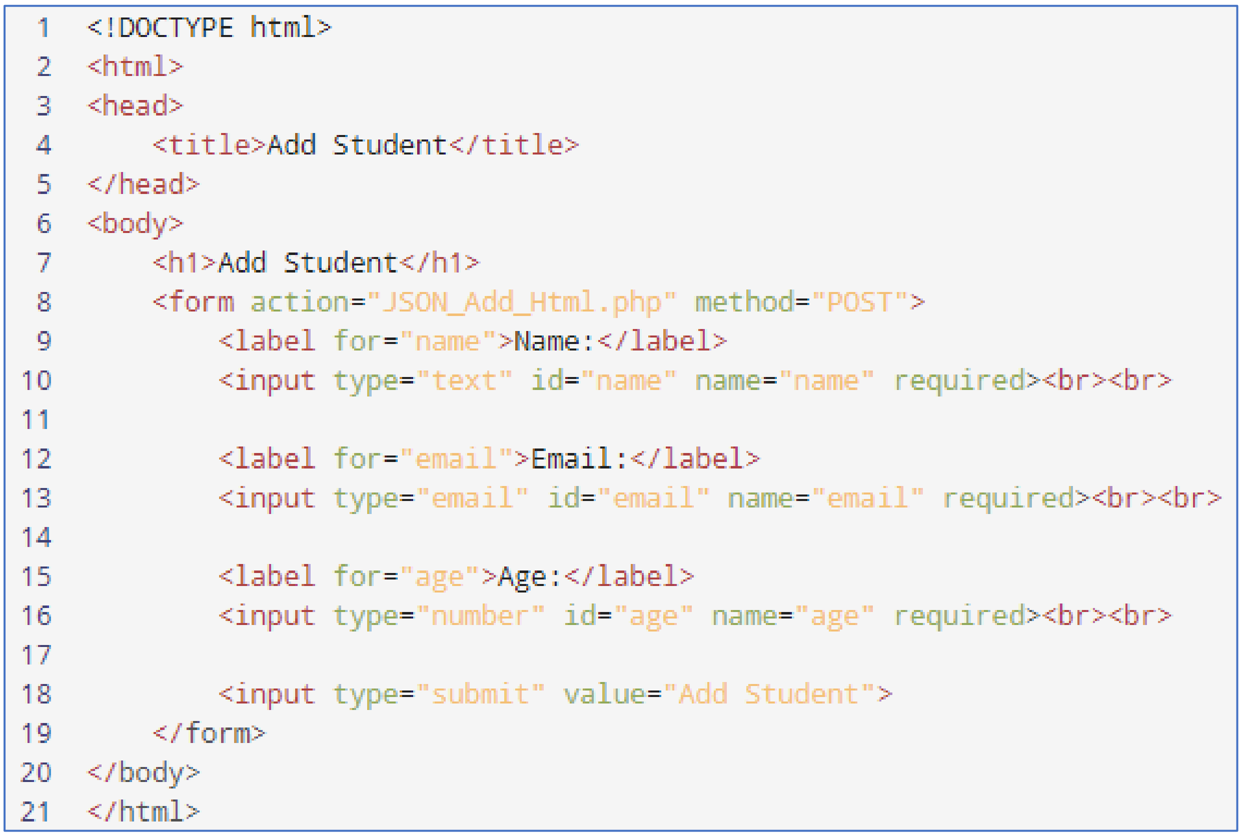
This line displays a success message indicating that the student data has been successfully exported to the CSV file.

<u>i. Adding Data to JSON Using HTML Form</u>

-> Query a user for student information using an HTML form and then save this to the JSON file considered in this lab sheet.

-> For this, you should create two new files called "JSON_Add.html" and "JSON_Add_Html.php".

Sample Code (HTML):



```html
<!DOCTYPE html>
<html>
<head>
    <title>Add Student</title>
</head>
<body>
    <h1>Add Student</h1>
    <form action="JSON_Add_Html.php" method="POST">
        <label for="name">Name:</label>
        <input type="text" id="name" name="name" required><br><br>

        <label for="email">Email:</label>
        <input type="email" id="email" name="email" required><br><br>

        <label for="age">Age:</label>
        <input type="number" id="age" name="age" required><br><br>

        <input type="submit" value="Add Student">
    </form>
</body>
</html>
```

Code Explanation (HTML):

```
<!DOCTYPE html>
<html>
<head>
<title>Add Student</title>
</head>
<body>
<h1>Add Student</h1>
```

Standard HTML Tags.

```
<form action="JSON_Add_Html.php" method="POST">
```

This line defines an HTML form and the **action** attribute specifies the URL or file to which the form data will be submitted while the **method** attribute specifies the HTTP method to be used (in this case POST).

```
<label for="name">Name:</label>
```

This line displays a label for the name input field. The for attribute associates the label with the corresponding input field using the id attribute.

```
<input type="text" id="name" name="name" required><br><br>
```

This line defines a text input field for the student's name. The id attribute uniquely identifies the input field and the name attribute specifies the name of the input field which will be used to access the value in PHP. The required attribute makes the field mandatory.

```
<label for="email">Email:</label>
<input type="email" id="email" name="email" required><br><br>
<label for="age">Age:</label>
<input type="number" id="age" name="age" required><br><br>
```

Similar to the previous step.

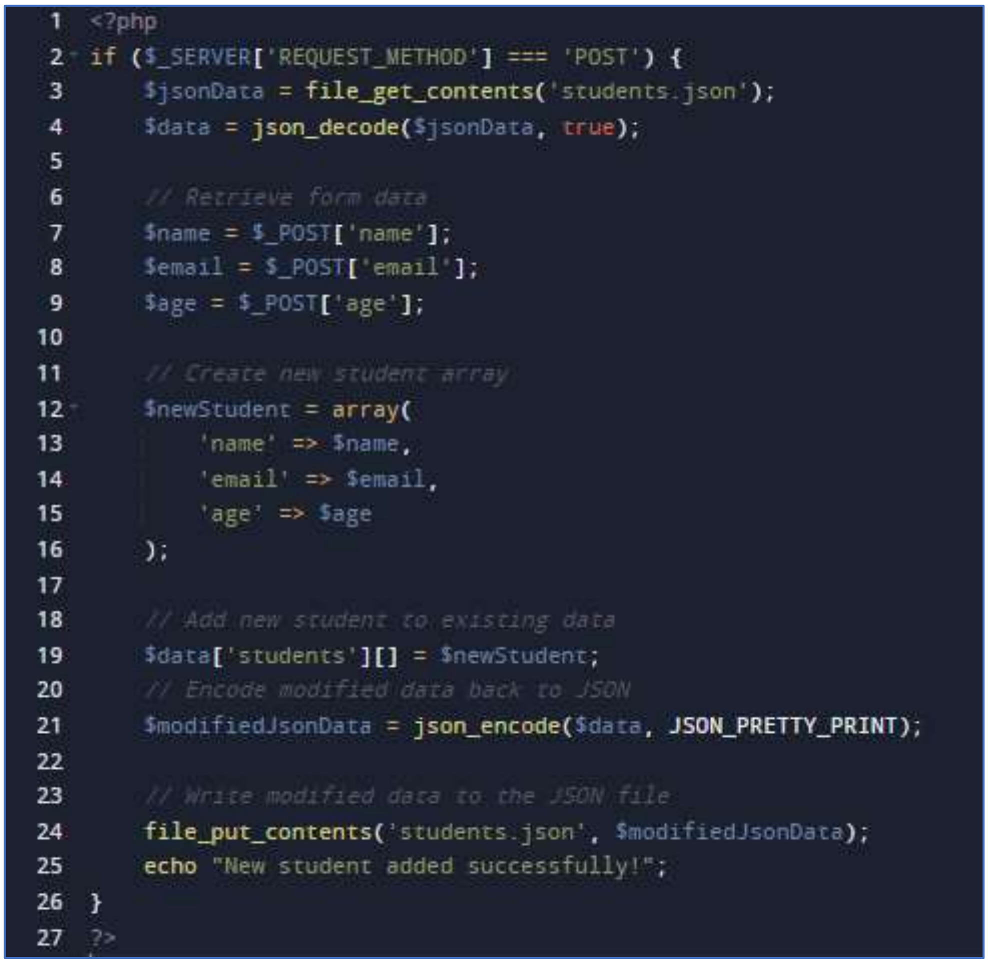
```
<input type="submit" value="Add Student">
```

This line creates a submit button. When clicked, it triggers the form submission and sends the entered data to the specified URL or file.

```
</form>
</body>
</html>
```

Standard HTML Closing Tags.

Sample Code (PHP):



```php
<?php
if ($_SERVER['REQUEST_METHOD'] === 'POST') {
    $jsonData = file_get_contents('students.json');
    $data = json_decode($jsonData, true);

    // Retrieve form data
    $name = $_POST['name'];
    $email = $_POST['email'];
    $age = $_POST['age'];

    // Create new student array
    $newStudent = array(
        'name' => $name,
        'email' => $email,
        'age' => $age
    );

    // Add new student to existing data
    $data['students'][] = $newStudent;
    // Encode modified data back to JSON
    $modifiedJsonData = json_encode($data, JSON_PRETTY_PRINT);

    // Write modified data to the JSON file
    file_put_contents('students.json', $modifiedJsonData);
    echo "New student added successfully!";
}
?>
```

Code Explanation (PHP):

if ($_SERVER['REQUEST_METHOD'] === 'POST') { ... }
This condition checks if the request method is **POST** indicating that the form has been submitted.

$jsonData = file_get_contents('students.json');**   **$data = json_decode($jsonData, true);
See previous examples for explanation on these lines of code.

$name = $_POST['name'];, $email = $_POST['email'];, $age = $_POST['age'];

These lines retrieve the form data submitted via **POST** method from the HTML form.

The **$_POST** superglobal array is used to access the values entered in the form fields using their respective name attributes.

$newStudent = array('name' => $name, 'email' => $email, 'age' => $age);

This line creates a new associative array named **$newStudent** to represent the data of the new student.

The values for the 'name', 'email', and 'age' keys are assigned from the form data retrieved in the previous step.

$data['students'][] = $newStudent;

This line adds the newly created student data in **$newStudent** to the 'students' array within the $data associative array.

The **[] notation** is used to append the new student data to the end of the **'students'** array.

$modifiedJsonData = json_encode($data, JSON_PRETTY_PRINT);

file_put_contents('students.json', $modifiedJsonData);

See previous examples for explanation on these lines of code.

echo "New student added successfully!";

This line displays a success message indicating that the new student has been added to the JSON file.