Criterion C

Tools and techniques

Backend:

- PHP - is a server-side scripting language that is widely used for web development.

- MySQL - is a relational database management system that is commonly used

- SSL (Secure Sockets Layer) encryption - is a security protocol that is used to establish a secure and encrypted connection between a web server and a web browser

Frontend:

- HTML - is the standard markup language used to create web pages.

- CSS - is used to style web pages and control the visual layout of elements.

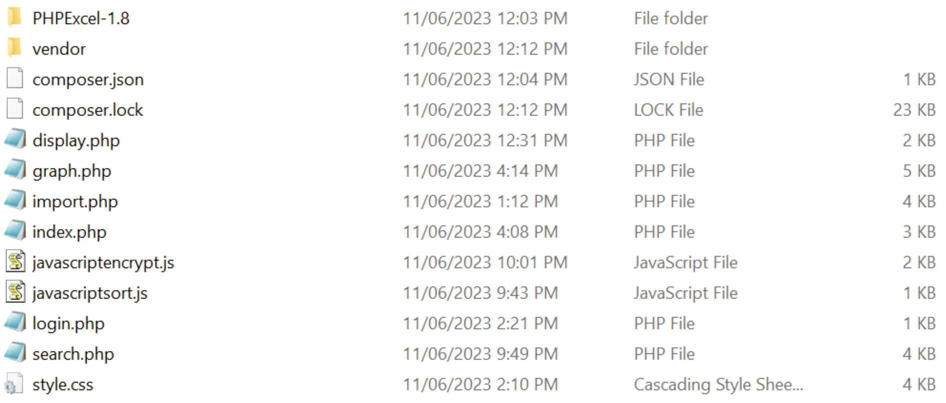
- JavaScript - is a programming language used to add interactivity and dynamic content to web pages.

- AJAX: - (Asynchronous JavaScript and XML) is a technique used to create fast and dynamic web pages without having to reload the entire page.

- jQuery - is a JavaScript library that simplifies the process of working with HTML documents and provides a variety of useful functions for frontend development.

- JGraph - is a JavaScript library for creating and displaying interactive diagrams and graphs. It provides a comprehensive set of tools and functionalities for visualising complex data in a graphical format.

Structure of the product

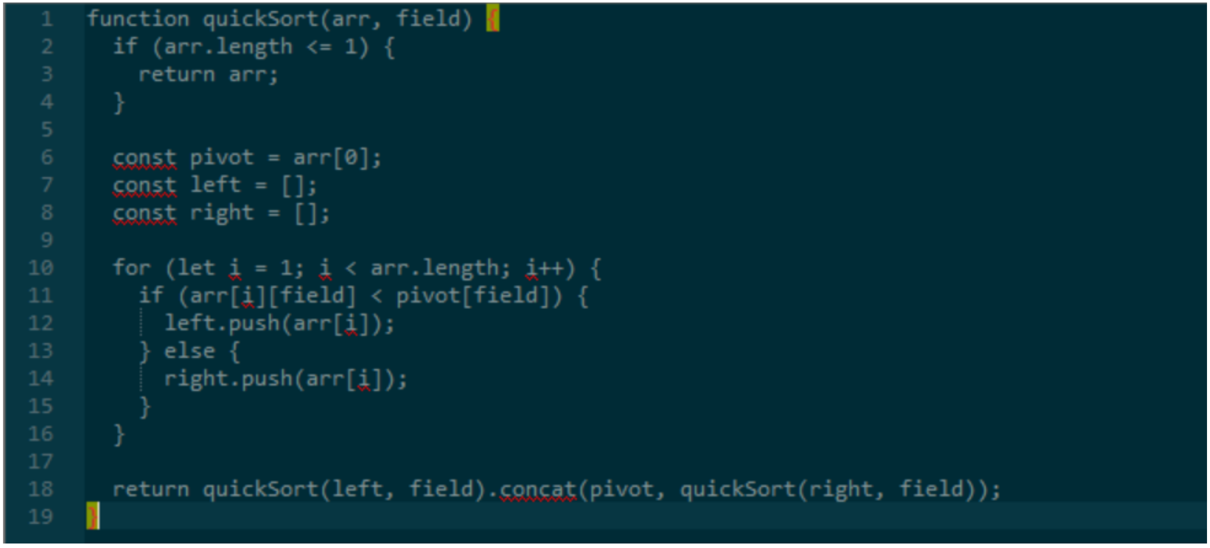


PHPExcel-1.8	11/06/2023 12:03 PM	File folder	
vendor	11/06/2023 12:12 PM	File folder	
composer.json	11/06/2023 12:04 PM	JSON File	1 KB
composer.lock	11/06/2023 12:12 PM	LOCK File	23 KB
display.php	11/06/2023 12:31 PM	PHP File	2 KB
graph.php	11/06/2023 4:14 PM	PHP File	5 KB
import.php	11/06/2023 1:12 PM	PHP File	4 KB
index.php	11/06/2023 4:08 PM	PHP File	3 KB
javascriptencrypt.js	11/06/2023 10:01 PM	JavaScript File	2 KB
javascriptsort.js	11/06/2023 9:43 PM	JavaScript File	1 KB
login.php	11/06/2023 2:21 PM	PHP File	1 KB
search.php	11/06/2023 9:49 PM	PHP File	4 KB
style.css	11/06/2023 2:10 PM	Cascading Style Shee...	4 KB

Figure 1: Data Structure of files (Further explanation Appendix A)

Tools and techniques

Quicksort algorithm

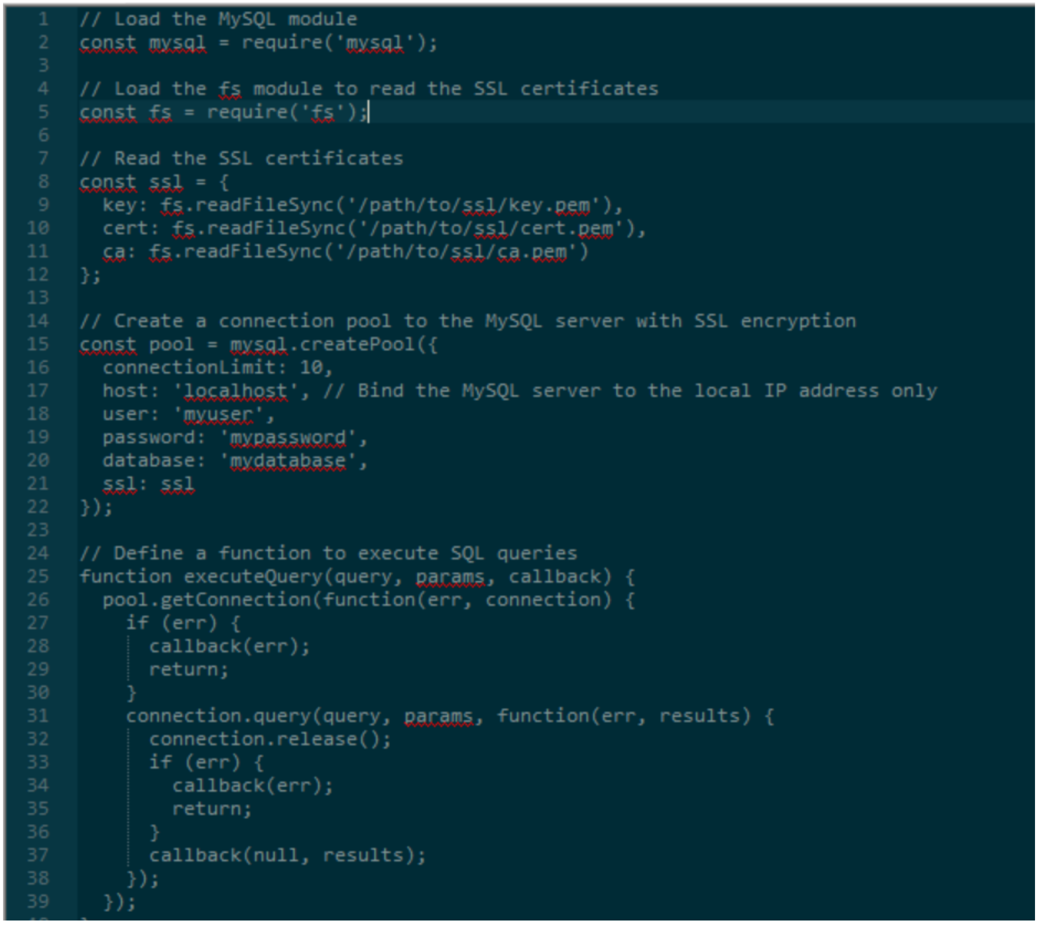


```
function quickSort(arr, field) {
  if (arr.length <= 1) {
    return arr;
  }

  const pivot = arr[0];
  const left = [];
  const right = [];

  for (let i = 1; i < arr.length; i++) {
    if (arr[i][field] < pivot[field]) {
      left.push(arr[i]);
    } else {
      right.push(arr[i]);
    }
  }

  return quickSort(left, field).concat(pivot, quickSort(right, field));
}
```

Figure 2: Quicksort Algorithm

As seen in figure 2 the quick Sort algorithm is a popular sorting algorithm that is widely used for its efficiency and speed, I specifically chose it for the website because it’s a recursive algorithm that sorts an array of elements by dividing it into two smaller sub-arrays, then recursively sorting these sub-arrays until they are sorted, recursion is specifically suited for my program as the database is made up of 1000+ entries meaning that it’s better to have larger problems sorted into smaller sections increasing efficiency and reducing runtime on the website. The quickSort() function above takes two arguments: an array of student grades and a field by which to sort the records. The function first checks if the array has only one or fewer elements, in which case it returns the array as it is already sorted. If the array has more than one element, the function selects the first element as the pivot point. The function then compares each element in the array to the pivot point based on the selected field. If the element is less than the pivot point, it is added to the left sub-array; otherwise, it is added to the right sub-array. The function then recursively calls itself on the left and right sub-arrays, concatenating the sorted left sub-array, pivot element, and sorted right sub-array into a final sorted array. This sorting algorithm is particularly useful for a school database management website as it can easily sort student records based on various fields, such as name, ID, age, or grades. furthermore, the quick sort algorithm has an average time complexity of O(n log n), which is fast and efficient compared to other algorithms such as bubble O(1) efficiency which makes my fitted quick sort algorithm a suitable algorithm for sorting large datasets. Overall, the quickSort() function. Its efficiency and flexibility make it an ideal choice for sorting data by different fields, and can improv efficiency throughout the whole website.

Encryption method

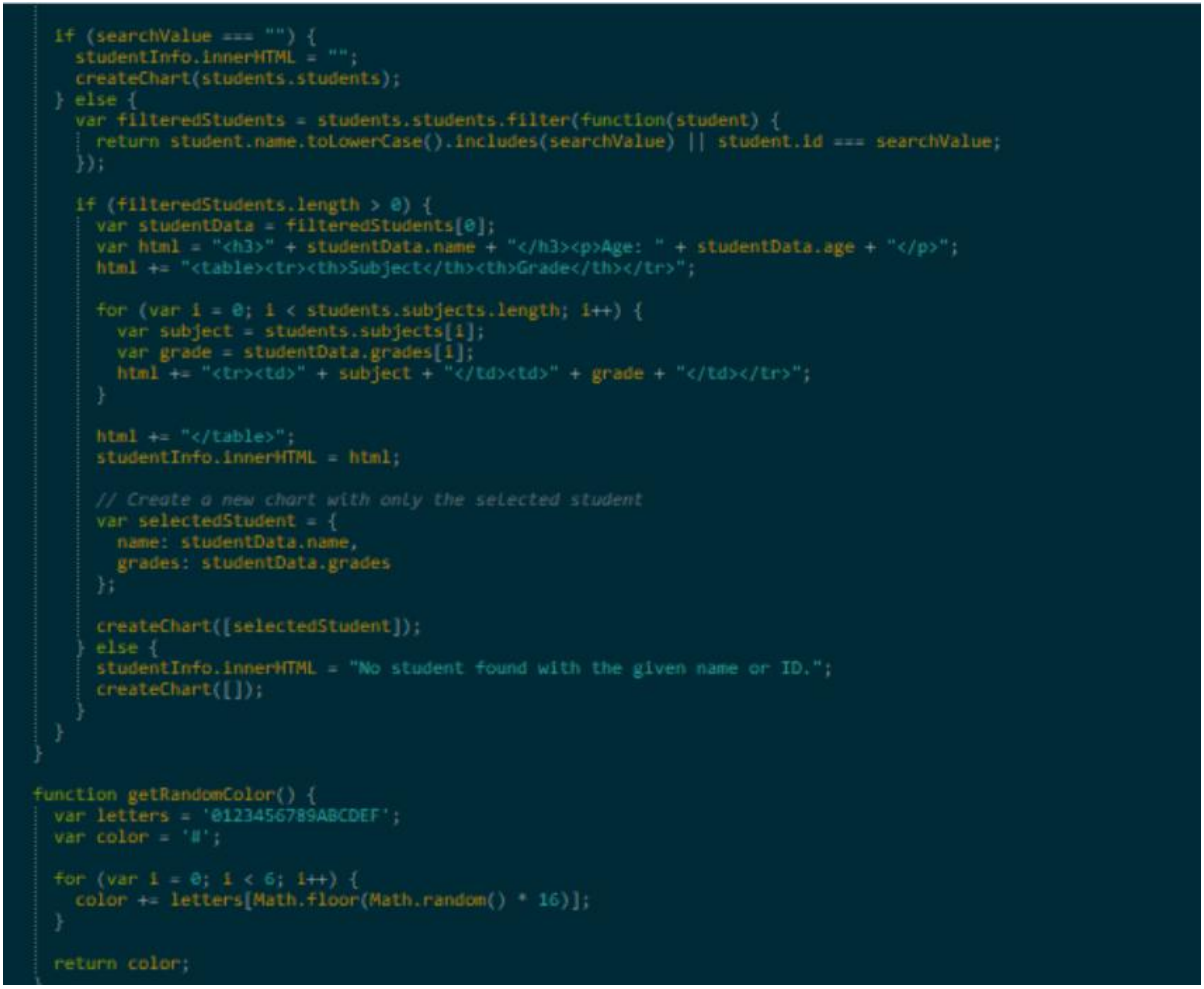


```
// Load the MySQL module
const mysql = require('mysql');

// Load the fs module to read the SSL certificates
const fs = require('fs');

// Read the SSL certificates
const ssl = {
  key: fs.readFileSync('/path/to/ssl/key.pem'),
  cert: fs.readFileSync('/path/to/ssl/cert.pem'),
  ca: fs.readFileSync('/path/to/ssl/ca.pem')
};

// Create a connection pool to the MySQL server with SSL encryption
const pool = mysql.createPool({
  connectionLimit: 10,
  host: 'localhost', // Bind the MySQL server to the local IP address only
  user: 'myuser',
  password: 'mypassword',
  database: 'mydatabase',
  ssl: ssl
});

// Define a function to execute SQL queries
function executeQuery(query, params, callback) {
  pool.getConnection(function(err, connection) {
    if (err) {
      callback(err);
      return;
    }
    connection.query(query, params, function(err, results) {
      connection.release();
      if (err) {
        callback(err);
        return;
      }
      callback(null, results);
    });
  });
```

Figure 3: Encryption Method

As seen in figure 3 the code for encryption implemented in the product demonstrates complexity and ingenuity. Firstly, the code utilises SSL encryption to establish a secure connection with the MySQL database. It loads the necessary SSL certificates from the file system, ensuring that the data transmitted between the application and the database is encrypted and protected from unauthorised access. The technique of using SSL encryption is appropriate for the product as it ensures the security and privacy of the client's data during transmission. By encrypting the data, it becomes significantly more challenging for unauthorised parties to intercept and decipher the information. The technique of SSL encryption meets the criteria for success related to data security and encryption. By implementing SSL encryption, the client's data is protected and kept secure during transmission between the web application and the MySQL database. This helps to maintain the confidentiality and integrity of the sensitive student information. Alternative techniques for securing data transmission include using VPN (Virtual Private Network) or IPsec (Internet Protocol Security) protocols. However, SSL/TLS (Transport Layer Security) encryption is widely supported and provides a robust and reliable method for securing data over the network. In the context of a student database management website, SSL encryption is the most appropriate choice as it is commonly used and provides a standardised approach to secure communication. Other tools or algorithms that could be considered for encryption include symmetric encryption algorithms (such as AES) or asymmetric encryption algorithms (such as RSA). However, in this case, SSL encryption is the preferred choice as it is specifically designed for secure communication between web applications and databases. It ensures end-to-end encryption without requiring the client to handle complex encryption and decryption processes. In summary, the implementation of SSL encryption in the code demonstrates complexity and ingenuity in securing the client's data during transmission. It meets the criteria for success related to data security and encryption, providing a reliable and standardised approach to protect sensitive information

Graph Function



```
  if (searchValue === "") {
    studentInfo.innerHTML = "";
    createChart(students.students);
  } else {
    var filteredStudents = students.students.filter(function(student) {
      return student.name.toLowerCase().includes(searchValue) || student.id === searchValue;
    });

    if (filteredStudents.length > 0) {
      var studentData = filteredStudents[0];
      var html = "<h3>" + studentData.name + "</h3><p>Age: " + studentData.age + "</p>";
      html += "<table><tr><th>Subject</th><th>Grade</th></tr>";

      for (var i = 0; i < students.subjects.length; i++) {
        var subject = students.subjects[i];
        var grade = studentData.grades[i];
        html += "<tr><td>" + subject + "</td><td>" + grade + "</td></tr>";
      }

      html += "</table>";
      studentInfo.innerHTML = html;

      // Create a new chart with only the selected student
      var selectedStudent = {
        name: studentData.name,
        grades: studentData.grades
      };

      createChart([selectedStudent]);
    } else {
      studentInfo.innerHTML = "No student found with the given name or ID.";
      createChart([]);
    }
  }
}

function getRandomColor() {
  var letters = '0123456789ABCDEF';
  var color = '#';

  for (var i = 0; i < 6; i++) {
    color += letters[Math.floor(Math.random() * 16)];
  }

  return color;
```

Figure 4: Graph function

As seen in figure 4 the implementation of dynamic graph generation in the student database management website was a complex and ingenious technique. The technique involved utilising the Chart.js library to generate interactive graphs based on the data fetched from the MySQL database. The code was implemented in such a way that it retrieved the necessary student data from the database, organised it, and then used Chart.js to create visually appealing and interactive graphs. The graphs were dynamically generated based on the selected student or all students, allowing users to easily analyse and compare student performance in different subjects. The technique was appropriate for the product as it provided a powerful visualisation tool that enhanced data analysis and decision-making. The graphs allowed users to quickly identify patterns, trends, and areas for improvement in student performance. This was crucial for teachers and administrators who needed to make informed decisions based on the data. The dynamic graph generation technique met several criteria for success. It provided an intuitive user interface where users could select specific students or view data for all students. The graphs were visually appealing, easy to understand, and allowed for interactive exploration by hovering over data points or adjusting the scales. The real-time updates ensured that the graphs reflected the most up-to-date data in the database. Alternative techniques, such as generating static images of the graphs or using server-side rendering, were considered but deemed less appropriate for the task. Static images would not have allowed for interactivity or real-time updates, limiting the usability and analytical capabilities of the application. Server-side rendering would have required frequent server requests and increased load times, resulting in a less responsive user experience.

Overall, the implementation of dynamic graph generation using Chart.js was an ingenious and complex technique that added significant value to the student database management website. It provided a powerful visualisation tool, the dynamic graph generation feature has met the criterion of success related to accurate data display , and outperformed alternative techniques in terms of interactivity, real-time updates, and user experience.

Existing tools

- XAMPP (Mac, Apache, MySQL, PHP) is a local web server environment that can be installed on a computer to create a development environment for web applications. It provides all the necessary components for building and testing web applications locally, including a web server, a database server, and a scripting language interpreter.

- PHPExcel is a PHP library that allows developers to work with Microsoft Excel files (.xls and .xlsx) using PHP. It provides various features and functions to create, read, modify, and manipulate Excel files programmatically. In the context of the product you mentioned, PHPExcel was used to import Excel files into the MySQL database. It allowed the client to easily import data from Excel spreadsheets, such as student records, and populate the corresponding database tables.