

Visualisation of IoT data

Author: Patrick Frank Atkinson (1811357)

ACDA5005 – Smart and Connected Devices I

Faculty of Architecture, Computing and Engineering

Date of submission: 28th February 2020 at 15:00

Abstract

This report outlines the author's implementation of a website for monitoring data collected from Internet of Things (IoT) devices. The report describes the functionality and technical aspects of the site on a page-by-page basis, and discusses the reasons why certain decisions were made with regards to design and implementation. The final section of the report reviews the implementation and suggests possible areas for improvement in the future.

Contents

1. Introduction.....	1
2. Implementation	1
2.1. Welcome.php.....	1
2.1.1. Functionality	1
2.1.2. Technical aspects	2
2.2. ConnInfo.php.....	2
2.3. DataAnalysis.php	3
2.3.1. Functionality	3
2.3.2. Technical aspect.....	3
2.4. ImportFromFiles.php.....	4
2.4.1. Functionality	4
2.4.2. Technical aspect.....	4
2.5. Login.php	5
2.5.1. Functionality	5
2.5.2. Technical aspect.....	6
2.6. Logout.php	6
2.6.1. Functionality	6
2.6.2. Technical details	6
2.7. TriggerDataImport.php	6
2.7.1. Functionality	6
2.7.2. Technical aspect.....	7
2.8. CheckCredentials.php	7
2.8.1. Functionality	7
2.8.2. Technical aspect.....	7
3. Conclusion	7
3.1. Further development	8

4.	Appendices.....	1
4.1.	Appendix A – ConnInfo.php.....	1
4.2.	Appendix B – Welcome.php	2
4.3.	Appendix C – Login.php.....	19
4.4.	Appendix D – Logout.php.....	21
4.5.	Appendix E – TriggerDataImport.php	23
4.6.	Appendix F – ImportFromFiles.php.....	26
4.7.	Appendix G – DataAnalysis.php.....	36
4.8.	Appendix H – CheckCredentials.php.....	57
4.9.	Appendix I – RainfallMonitor.js	59

1. Introduction

The brief requires a website be created that imports rainfall depth data from sample text files into a database. The data should be presented appropriately to the end user, with charts and statistics to assist in interpreting the data. They should also be able filter the data as they see fit. Security measures should be in place to secure the data from unauthorised access.

2. Implementation

The website is made up of eight PHP pages. The following section will provide details of the functionality (where applicable) and technical aspects of each page.

2.1. Welcome.php

2.1.1. Functionality

As its name implies, the welcome page is the landing page for the site. Users that have not yet authenticated see a welcome message and are informed of how to log in. Authenticated users see a different message that informs them of the functionality available on the site. If there is rainfall data in the database, both groups of users are presented with some overview statistics based on the data, such as maximum and average rainfalls. A pie chart is also displayed that shows the total rainfall depth for each sensor on the last day with readings in the database. A pie chart was chosen as it helps with comparison between the sensors.

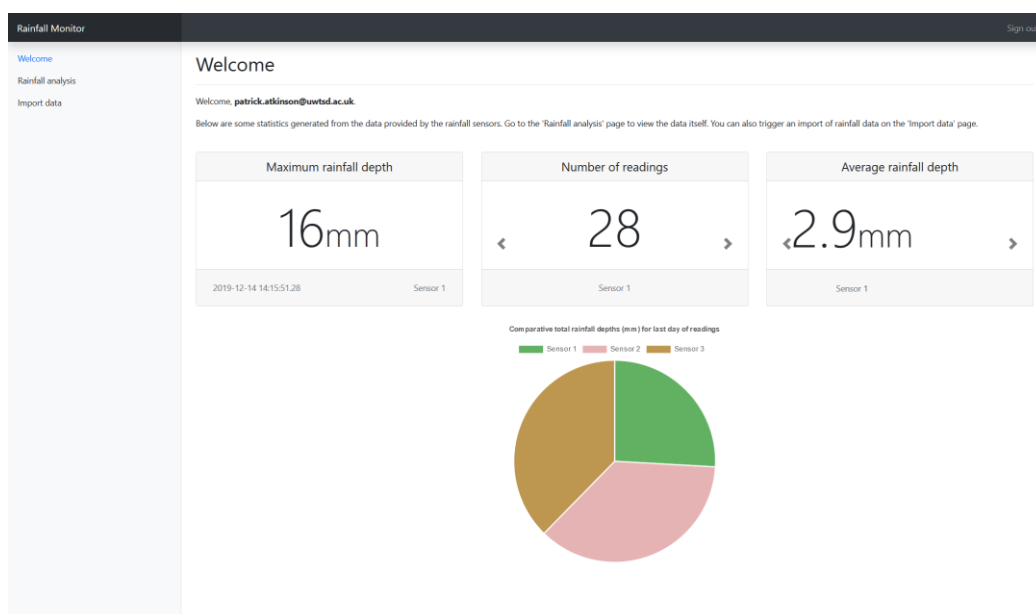


Figure 1: Screenshot of the Welcome.php page as an authenticated user would see it. This shows the overview statistics on the rainfall data.

2.1.2. Technical aspects

The code for this page can be found in Appendix B – Welcome.php. The welcome page is based on the ‘Dashboard’ template available on getbootstrap.com. This template was chosen as it reduced the time required for implementing a layout that is responsive and suited to displaying data. Unwanted elements and references were removed to simplify the markup of the page. The majority of the PHP code is at the start of the markup in an attempt to keep it together for ease of maintenance.

There is a section at the start of the PHP code that checks the session cookies to confirm whether the user of the page has authenticated, and subsequently sets a variable. The variable is used later in the markup to control the messages displayed to the user. It also controls the display of the navigation and whether the log-in or log-out link is displayed.

The remaining PHP at the start of the markup is used for querying the database for the rainfall statistics that are presented on the page. The first query is executed. If this indicates that there is no rainfall data in the database, the user is informed of the lack of data on-screen and no statistics are displayed. Otherwise, the remaining queries are executed, and a variable is created for each set of results. All queries use the `mysqli_query` function. This function is used as there is no user interaction with the query and therefore protection against SQL injection is less critical. The result sets are looped through later in the markup and the values are incorporated into the HTML.

The pie chart is generated in the same way as the charts on DataAnalysis.php. Further details can be found in the Technical aspects section for that page later in this report.

2.2. ConnInfo.php

The code for this page can be found in Appendix A – ConnInfo.php. This file creates a connection to the MySQL database. This file is referenced by most of the other PHP files in the website when they require a connection to the database. This reduces duplication of code across the pages and makes maintenance simpler if the database details change.

2.3. DataAnalysis.php

2.3.1. Functionality

This page presents the raw rainfall data to the user, along with charts to assist in visualising and analysing the data. The charts chosen show rainfall depths for the sensors over time as either a line or bar chart, along with rainfall statistics (average, maximum and minimum depths) over time as a line chart. The line chart for rainfall depth can help visualise trends for rainfall. The bar chart can help visualise the total rainfall depth for each point in time, as well as how the rainfall depth compares between the sensors. The data can be filtered as necessary, with the filters being applied to the table of data and the charts. There are also pre-set filters available that allow the user to quickly filter to a particular timeframe.

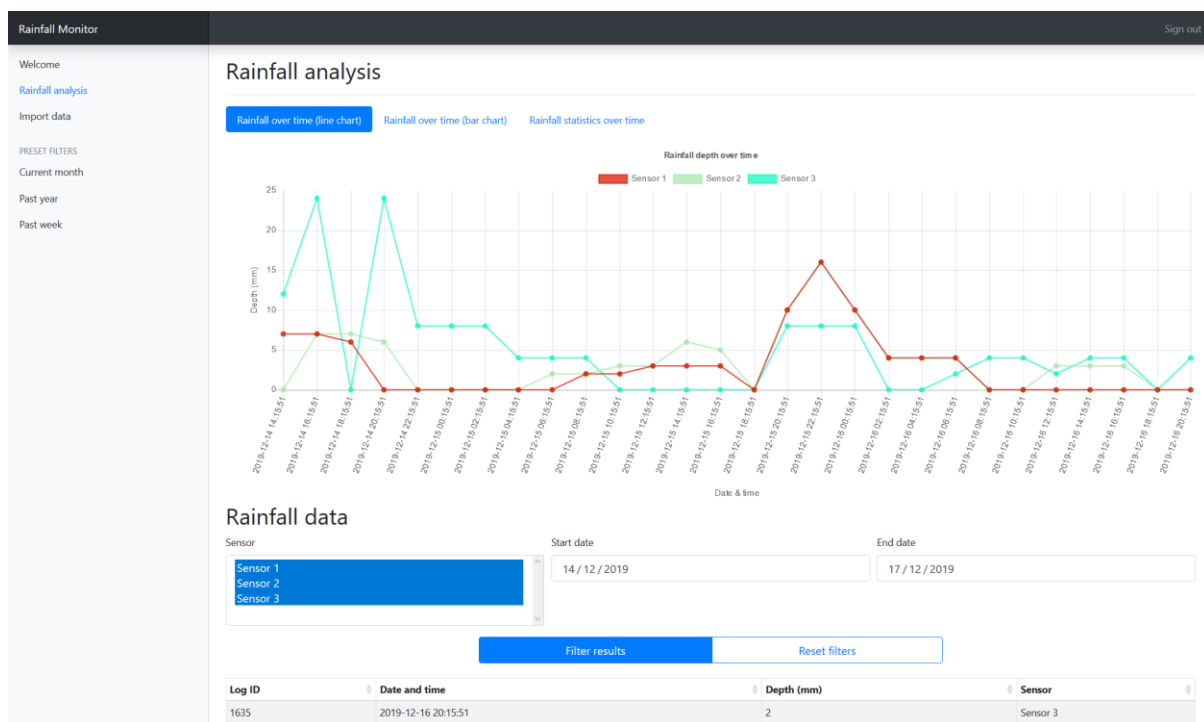


Figure 2: Screenshot of the DataAnalysis.php page showing the chart and the form used for filtering the data.

2.3.2. Technical aspect

The code for this page can be found in Appendix G – DataAnalysis.php. This page is based on the same template as Welcome.php. An initial check is made to ensure the user is authenticated. They are redirected to Login.php if not, otherwise the remaining code executes. There are functions used to generate values for the fields in the HTML form based on data in the rainfall log table. POSTed data is subsequently added to variables if present. A check is carried out to ensure there is rainfall data present in the database. If not, the user is informed, and no further

queries are executed. If data is present, the POSTed data and values from the previously mentioned functions are used to build the HTML form inputs.

A query is executed that returns rainfall data based on the form inputs. All the rainfall data is returned by default. The data is formatted as a HTML table. Two further queries are executed with the same WHERE clause. These query results are encoded as JSON as required by the chart library. The charts are constructed using Chart.js. This uses JavaScript to generate HTML5 charts and add them to specific elements on the page. The JSON for each chart is ‘echoed’ into the JavaScript and a function is called to add each chart to the page. Chart.js was chosen as it allows multiple datasets on the same chart which is useful for comparisons. The charts are interactive in that the user can easily see the value for each data point by hovering on it. It can also scale appropriately based on viewport.

JavaScript is also used for the pre-set filters. The JavaScript calculates the appropriate date range, sets the form inputs to those values, and submits the form. The JavaScript file can be found in Appendix I – RainfallMonitor.js.

2.4. ImportFromFiles.php

2.4.1. Functionality

This page imports the rainfall data that is held in text files into the MySQL database. The results of the import are displayed to the user. This includes details of which files were imported, how many rows in each file were imported and error messages if there were issues. Rows of data that cannot be imported are added to a separate text file for later review.

2.4.2. Technical aspect

The code for this page can be found in Appendix F – ImportFromFiles.php. This page is based on the same template as Welcome.php. An array of filenames is retrieved from the database with an SQL query and each file is opened in turn. This approach was chosen as it does not rely on the filename to identify the sensor. The rows of data in the files are iterated over, with the data from the row being added to variables. The SQL INSERT statement declared previously in the PHP is used with functions from the mysqli_stmt class to pass the data to the statement as parameters and subsequently execute the statement. This is repeated for each row. The mysqli_stmt class is used instead of the mysqli_query function to prevent SQL injection from occurring.

A check is carried out after the statement execution to see whether it was successful. If so, the count of successful imports is incremented. If not, an alert is printed to the screen for that particular row and the row data (including the source filename) is appended to a separate text file called failedImports.txt. This ensures the data is not lost when the rainfall files are truncated at the end of the import.

After a file has been processed, the user is informed of the number of successfully imported rows, and the file is truncated to ensure the same data is not reimported in the future. Error alerts are displayed in the order they are encountered. A further warning message regarding the failedImports.txt file is displayed if errors did occur.

The screenshot shows a web interface titled "Import results". It displays a sequence of status messages: "Beginning import of file rainfall1.txt...", "Successfully imported 28 of 28 rows.", "Beginning import of file rainfall2.txt...", "Successfully imported 28 of 28 rows.", "Beginning import of file rainfall3.txt...", and "Successfully imported 28 of 28 rows.".

Figure 3: Example of a successful import

The screenshot shows a web interface titled "Import results". It displays the message "Beginning import of file rainfall4.txt...". Below this, a red error box contains the text: "Error running query on row 9 with following values: 2, 2019-12-15 h06:15:51.282545, 4." Below the error box, it says "Successfully imported 27 of 28 rows." At the bottom, another red box states: "1 rows failed to import into the database. They have been written to the file **failedImports.txt**."

Figure 4: Example of an import that had an error

2.5. Login.php

2.5.1. Functionality

The log-in page allows the user to authenticate and gain access to the rest of the website.

The screenshot shows a web interface titled "Rainfall Monitor" with the subtitle "Please sign in". It features two input fields: "Email address" and "Password". Below these fields is a blue button labeled "Sign in".

Figure 5: Screenshot of the form on Login.php.

The screenshot shows the same login interface as Figure 5, but with an additional red error box at the top that reads: "Invalid credentials entered. Please try again." The "Email address" field now contains the text "patrick.atkinson@uwtsd.ac.uk". The "Password" field is empty. The "Sign in" button remains at the bottom.

Figure 6: Screenshot of the form on Login.php after the user has failed to authenticate successfully.

2.5.2. Technical aspect

The code for this page can be found in Appendix C – Login.php. The log-in page is based on the ‘Sign-in’ template from getbootstrap.com, chosen for similar reasons to the ‘Dashboard’ template. In the PHP, a check is made to see if the page has been POSTed to. If not, two variables are set as blank. If data has been POSTed, the variables are set to that data. Data will only be POSTed to Login.php if the user has previously failed to authenticate. In this scenario, an error message is displayed informing the user of the failed authentication. The email address they entered is also added back into the relevant form input. The form prompts the user for their email address and password. The form cannot be submitted if an invalid email address is entered or if no password is entered. The data from the form is POSTed to the CheckCredentials.php page on submission.

2.6. Logout.php

2.6.1. Functionality

The log-out page is a straight-forward page that logs the user out, informs them they have been logged out and redirects them to the welcome page after three seconds.

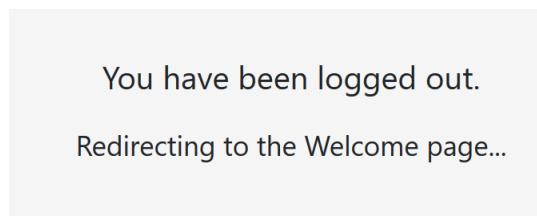


Figure 7: Screenshot of the message displayed to the user when they log out.

2.6.2. Technical details

The code for this page can be found in Appendix D – Logout.php. The current session is resumed. It is then ‘destroyed’ to remove the username session cookie that indicates the user is logged in. The user is informed they have been logged out and is subsequently redirected to Welcome.php after three seconds.

2.7. TriggerDataImport.php

2.7.1. Functionality

This page allows the user to trigger an import of the rainfall data that is held in text files.

2.7.2. Technical aspect

The code for this page can be found in Appendix E – TriggerDataImport.php. This page is based on the same template as Welcome.php. The ‘Import data’ button redirects the user to ImportFromFiles.php.

2.8. CheckCredentials.php

2.8.1. Functionality

This page is not displayed to the user. It redirects the user based on the outcome of the authentication check. The user is redirected to the welcome page if their credentials were correct, otherwise they are returned to the log-in page.

2.8.2. Technical aspect

The code for this page can be found in Appendix H – CheckCredentials.php. This page takes the credentials POSTed from Login.php and passes them as parameters to a statement defined using functions from the mysqli_stmt class. The statement is executed, and a variable is set with a count of the number of rows returned by the query. A single row is returned if the credentials are correct, in which case a username session cookie is set with the username previously entered by the user, and the user is redirected to Welcome.php.

If zero or multiple rows are returned by the query, a hidden form that POSTs to Login.php is created. This contains the username entered and a value to indicate the credentials were incorrect. This form is submitted automatically with JavaScript.

3. Conclusion

A key factor for the development of the software was to ensure it is flexible enough to account for expanded use of the system. The search options and display of the data (both text and visualised) both expand as more rainfall data and sensors are added to the system. Another focus of the development was to put measures in place for error checking, such as when the database is empty or when there is an issue importing data into the database. This was done to provide more ‘elegant’ error messages that inform the user of the issue in plain English rather than typical PHP exception messages.

3.1. Further development

A further development of the website could be to alter the import process to write details of failed row imports to a database table rather than a text file, and provide functionality to review, correct and re-import that data.

Additional functionality could also be added that allows the user to add new sensors to the database, as this currently requires the table to be updated through the Database Management System. Similar functionality could also be added for creating new user accounts.

4. Appendices

4.1. Appendix A – ConnInfo.php

```
<?php
    $strServer = "127.0.0.1";
    $strDatabase = "1811357";      //      Database name
    $strUser = "root";
    $strPwd = "password";          //      MySQL password

    $link = mysqli_connect($strServer, $strUser, $strPwd, $strDatabase) or die("Could not open database");
?>
```

4.2. Appendix B – Welcome.php

```
<?php

#####
##
//~~~~~
~~
//      Section for checking whether the user is logged and setting a variable with the result
//~~~~~
~~

//      Create a new session or resume the current one
session_start();

//      Set a variable with the value of the 'username' session cookie
$user = @$_SESSION["username"];

//      If the variable is blank (i.e. the user isn't logged in), a variable is set to false, otherwise it is
set to
//      true
if ($user == "")
{
    $authenticatedUser = false;
}
else
{
    $authenticatedUser = true;
}
#####
##

#####
##
//~~~~~
~~
//      Function for returning encoded JSON based on the passed in query and date parameters
```

```

//~~~~~
~~

function ReturnChartJSON(string $SQLQuery)
{
    include "ConnInfo.php";

    //    Initialise a new statement with the connection set in the ConnInfo.php file
    $stmt = mysqli_stmt_init($link);

    //    Prepare the statement with the query
    mysqli_stmt_prepare($stmt, $SQLQuery);

    //    Execute the statement and return the result
    $result = mysqli_stmt_execute($stmt);

    //    Set variable with the data from the query
    $data = mysqli_stmt_get_result($stmt);

    //    Check for rows
    if (mysqli_num_rows($data) > 0)
    {
        $chartData = array();

        //    Add the chart data to an array
        while ($r = mysqli_fetch_assoc($data))
        {
            $chartData[] = $r;
        }

        //    Encode the array of chart data as JSON
        $chartJSON = json_encode($chartData, JSON_PRETTY_PRINT);

        return $chartJSON;
    }
    else
    {
        //    No rows returned for query
        return "";
    }
}

```

```

    }
}

//#####

##

//#####

##
//~~~~~
~~
//    Section for querying the database for the maximum rainfall depth
//~~~~~
~~

include "ConnInfo.php";

$querySelect = "  SELECT
                    rf.LogDateTime
                    ,    MAX(rf.depth)      AS    MaxDepth
                    ,    s.SensorName
                FROM    rainfalllog rf
                INNER JOIN  sensors s          ON    rf.SensorID = s.sensorID";

//    Set variable with the data from the query
$result = mysqli_query($link, $querySelect);

//    Set variable with the row count from the query
$rowCount = mysqli_num_rows($result);

$maxDepthDateTime = "";
$maxDepth = "";
$maxDepthSensor = "";

//    Check for rows
if ($rowCount > 0)
{

```

```

//      Iterate through array of query results and set variables with the field values from the query
while ($r = mysqli_fetch_array($result))
{
    $maxDepthDateTime = $r["LogDateTime"];
    $maxDepth = $r["MaxDepth"];
    $maxDepthSensor = $r["SensorName"];
}

}

if ($maxDepthDateTime === null)
{
    $returnMessageNoData = "No rainfall data in the database.";
}

//#####

##

//      Check to see whether $returnMessageNoData has been set (it is only set if there is no rainfall data)
if (!isset($returnMessageNoData))
{

//#####

##

//~~~~~

~~

    //      Section for querying the database for the number of readings and average rainfall per sensor

//~~~~~

~~

    include "ConnInfo.php";

    $querySelect = "  SELECT

                                COUNT(rf.depth)    AS    ReadingCount
                                ,    s.SensorName

                                FROM  rainfalllog rf

```

```

INNER JOIN  sensors s          ON    rf.SensorID = s.sensorID

GROUP BY
        s.SensorName";

//      Set variable with the data from the query
$resultReadingCounts = mysqli_query($link, $querySelect);

$querySelect = "  SELECT
                ROUND(AVG(rf.depth), 1) AS    AverageDepth
            ,
                s.SensorName

FROM    rainfalllog rf

INNER JOIN  sensors s          ON    rf.SensorID = s.sensorID

GROUP BY
        s.SensorName";

//      Set variable with the data from the query
$resultAverageDepth = mysqli_query($link, $querySelect);

$querySelect = "  SELECT
                s.SensorID

FROM    sensors s";

//      Set variable with the data from the query
$resultSensorCount = mysqli_query($link, $querySelect);

//      Set variable with a count of the rows (i.e. the number of sesnors)
$sensorCount = mysqli_num_rows($resultSensorCount);

```

```

//#####

```

```

##

```

```

//#####
##

//~~~~~
~~
//      Section for querying the database for rainfall statistics and populating a variable with the
returned data
//      encoded as JSON to be used for the Chart.js

//~~~~~
~~

$query = "  SELECT
                s.SensorName
            ,    SUM(rf.Depth)          AS    Depth
            FROM
                rainfalllog rf
            INNER JOIN  sensors s          ON    rf.SensorID = s.SensorID
            GROUP BY
                DATE(rf.LogDateTime), rf.SensorID

            ORDER BY
                DATE(rf.LogDateTime) DESC, rf.SensorID ASC

            LIMIT $sensorCount;";

//      Call the ReturnChartJSON function to get the JSON encoded result of the above query
$dailyStatisticsChartJSON = ReturnChartJSON($query);

//#####
##
}

?>

```

```

<!doctype html>
<html lang="en">
  <head>

    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
    <title>Rainfall Monitor - Welcome</title>

    <!-- Bootstrap core CSS -->
    <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/css/bootstrap.min.css"
integrity="sha384-Vkoo8x4CGsO3+Hhxv8T/Q5PaXtkKtu6ug5TOeNV6gBiFeWPGFN9MuhOf23Q9Ifjh" crossorigin="anonymous">

    <!-- Template specific CSS -->
    <link rel="stylesheet" href="https://getbootstrap.com/docs/4.4/examples/dashboard/dashboard.css">

    <style>

      .carousel-indicators li {
        background-color: #000000;
      }

      .carousel-control-next,
      .carousel-control-prev {
        filter: invert(100%);
      }

    </style>

  </head>

  <body>

    <nav class="navbar navbar-dark fixed-top bg-dark flex-md-nowrap p-0 shadow">
      <a class="navbar-brand col-sm-3 col-md-2 mr-0" href="Welcome.php">Rainfall Monitor</a>

      <ul class="navbar-nav px-3">
        <li class="nav-item text-nowrap">
          <?php

```

```

        //      Display different HTML depending on whether or not the user is logged in
        if ($authenticatedUser == false)
        {
            echo "<a class=\"nav-link\" href=\"Login.php\">Sign in</a>";
        }
        else
        {
            echo "<a class=\"nav-link\" href=\"Logout.php\">Sign out</a>";
        }
    ?>
</li>
</ul>
</nav>

<div class="container-fluid">

    <div class="row">

        <nav class="col-md-2 d-none d-md-block bg-light sidebar">
            <div class="sidebar-sticky">

                <ul class="nav flex-column">

                    <li class="nav-item">
                        <a class="nav-link active" href="Welcome.php">
                            Welcome <span class="sr-only">(current)</span>
                        </a>
                    </li>

                    <?php
                        //      Only display the navigation links if the user is authenticated
                        if ($authenticatedUser === true)
                        {
                            ?>
                            <li class="nav-item">
                                <a class="nav-link" href="DataAnalysis.php">
                                    Rainfall analysis
                                </a>
                            </li>

```

```

        <li class="nav-item">
            <a class="nav-link" href="TriggerDataImport.php">
                Import data
            </a>
        </li>

        <?php
        }
        ?>
    </ul>

</div>
</nav>

<main role="main" class="col-md-9 ml-sm-auto col-lg-10 px-4">

    <div class="d-flex justify-content-between flex-wrap flex-md-nowrap align-items-center
pt-3 pb-2 mb-3 border-bottom">
        <h1 class="h2">Welcome</h1>
    </div>

    <?php
    //      Display different HTML depending on whether or not the user is logged in
    if ($authenticatedUser == false)
    {
        echo "<p>Welcome to the Rainfall Monitor site.</p>";

        //      Display a different message if there is no data in the database
        if (!isset($returnMessageNoData))
        {
            echo "<p>
rainfall sensors.
                You will need to log in to view the data itself. You can log in by
clicking 'Sign in' at the top of the screen.
            </p>";
        }
        else

```

```

        {
            echo "<p>
                There is currently no rainfall data in the database. You will need
to log in to import data.
                You can log in by clicking 'Sign in' at the top of the screen.
            </p>";
        }
    }
    else
    {
        echo "<p>Welcome, <b>$user</b>.</p>";

        //    Display a different message if there is no data in the database
        if (!isset($returnMessageNoData))
        {
            echo "<p>
                Below are some statistics generated from the data provided by the
rainfall sensors.
                Go to the 'Rainfall analysis' page to view the data itself. You can
also trigger an import of rainfall data on the
                'Import data' page.
            </p>";
        }
        else
        {
            echo "<p>
                There is currently no rainfall data in the database. You can also
trigger an import of rainfall data on the
                'Import data' page.
            </p>";
        }
    }
?>
<br>

<?php
//    Display the statistics only if $returnMessageNoData is not set (i.e. there is
data in the database)
if (!isset($returnMessageNoData))

```

```

{
?>
<div class="card-deck" id="overviewDataCards">

    <div class="card text-center">
        <div>
            <h5 class="card-header">Maximum rainfall depth</h5>
        </div>

        <div class="card-body">
            <span class="display-2"><?php
                echo $maxDepth;
            ?></span><span class="display-4">mm</span>
        </div>

        <div class="card-footer text-muted">
            <div class="d-flex flex-row justify-content-between">
                <div class="p-2">
                    <?php
                        echo $maxDepthDateTime;
                    ?>
                </div>

                <div class="p-2">
                    <?php
                        echo $maxDepthSensor;
                    ?>
                </div>
            </div>
        </div>
    </div>

    <div class="card text-center">
        <div>
            <h5 class="card-header">Number of readings</h5>
        </div>

        <div id="readingsCarousel" class="carousel slide" data-ride="carousel">

```

on-screen

the first time through the loop

content-center">

```
<div class="carousel-inner">
<?php

    //      Set variable with the row count from the query
    $rowCount = mysqli_num_rows($resultReadingCounts);

    $loopCount = 0;

    //      Check for rows
    if ($rowCount > 0)
    {
        //      Iterate through array of query results and display

        while ($r = mysqli_fetch_array($resultReadingCounts))
        {
            //      Add the 'active' class in the HTML if it is

            if ($loopCount == 0)
            {
                echo "<div class=\"carousel-item active\">";
            }
            else
            {
                echo "<div class=\"carousel-item\">";
            }

            <div class="card-body">
                <span class="display-2"><?php
                    echo $r["ReadingCount"];
                ?></span>
            </div>

            <div class="card-footer text-muted">
                <div class="d-flex flex-row justify-

                    <div class="p-2">
                        <?php
```

```

$ r["SensorName"];

                                                                    echo
                                                                    ?>
                                                                    </div>
                                                                    </div>
                                                                    </div>
                                                                    </div>
                                                                    </div>
<?php
    //      Increment the loop counter
    $loopCount++;
}
?>
</div>

role="button" data-slide="prev">
hidden="true"></span>

role="button" data-slide="next">
hidden="true"></span>

<a class="carousel-control-prev" href="#readingsCarousel"
    <span class="carousel-control-prev-icon" aria-
    <span class="sr-only">Previous</span>
</a>

<a class="carousel-control-next" href="#readingsCarousel"
    <span class="carousel-control-next-icon" aria-
    <span class="sr-only">Next</span>
</a>

</div>
</div>
<div class="card text-center">
    <div>
        <h5 class="card-header">Average rainfall depth</h5>
    </div>

```

within the HTML

the first time through the loop

4">mm

```
<div id="averagesCarousel" class="carousel slide" data-ride="carousel">

  <div class="carousel-inner" style="height: 100%;">
    <?php

      //      Set variable with the row count from the query
      $rowCount = mysqli_num_rows($resultAverageDepth);

      $loopCount = 0;

      //      Check for rows
      if ($rowCount > 0)
      {
        //      Iterate through array of query results and output

        while ($r = mysqli_fetch_array($resultAverageDepth))
        {
          //      Add the 'active' class in the HTML if it is

          if ($loopCount == 0)
          {
            echo "<div class=\"carousel-item active\">";
          }
          else
          {
            echo "<div class=\"carousel-item\">";
          }

          <div class="card-body">
            <span class="display-2"><?php
              echo $r["AverageDepth"];
            ?></span><span class="display-
          </div>

          <div class="card-footer text-muted">
```

```

content-center">
    <div class="d-flex flex-row justify-
        <div class="p-2">
            <?php
                echo
            ?>
        </div>
    </div>
</div>
<?php
    // Increment the loop counter
    $loopCount++;
}
?>
</div>
<a class="carousel-control-prev" href="#averagesCarousel"
    <span class="carousel-control-prev-icon" aria-
        <span class="sr-only">Previous</span>
    </a>
<a class="carousel-control-next" href="#averagesCarousel"
    <span class="carousel-control-next-icon" aria-
        <span class="sr-only">Next</span>
    </a>
</div>
</div>

```

```

        </div>

        <br>

        <div style="height: 40vh;">
            <canvas id="dailyStatisticsChart"></canvas>
        </div>

        <?php
        }
        ?>

    </main>

</div>
</div>

    <script src="https://code.jquery.com/jquery-3.4.1.slim.min.js" integrity="sha384-
J6qa4849blE2+poT4WnyKhv5vZF5SrPo0iEjwBvKU7imGFAV0wwjlYyfoRSJoZ+n" crossorigin="anonymous"></script>
    <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/js/bootstrap.min.js" integrity="sha384-
wfSDF2E50Y2D1luUdj003uMBJnjuUD4Ih7YwaYdliqfktj0Uod8GCExl3Og8ifwB6" crossorigin="anonymous"></script>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/Chart.js/2.7.3/Chart.min.js"></script>
    <script src="js/RainfallMonitor.js"></script>

    <script>

        //      Set variable with the element the chart will be added to
        var dailyStatisticsChartElement = document.getElementById('dailyStatisticsChart');

        //      Set variable with the JSON generated by the PHP
        var dailyStatisticsChartJSON = <?php
        echo $dailyStatisticsChartJSON;
        ?>

        //      Call the CreateBarChart function (see RainfallMonitor.js) for the chart
        CreatePieChart(dailyStatisticsChartElement, dailyStatisticsChartJSON);

    </script>

```

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

4.3. Appendix C – Login.php

```
<?php
//      Check to see if anything has been posted to the page. Sets variables as necessary
if (empty($_POST))
{
    $access = "";
    $email = "";
}
else
{
    $access = $_POST["access"];
    $email = $_POST["email"];
}
?>

<!doctype html>
<html lang="en">
    <head>

        <meta charset="utf-8">
        <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
        <title>Rainfall Monitor - Login</title>

        <!-- Bootstrap core CSS -->
        <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/css/bootstrap.min.css"
integrity="sha384-Vkoo8x4CGsO3+Hhvx8T/Q5PaXtkKtu6ug5TOeNV6gBiFeWPGFN9MuhOf23Q9Ifjh" crossorigin="anonymous">

        <!-- Custom styles for this template -->
        <link href="css/signin.css" rel="stylesheet">

    </head>

    <body class="text-center">

        <form class="form-signin" action="CheckCredentials.php" method="POST">

            <h1>Rainfall Monitor</h1>
```

```

<h1 class="h3 mb-3 font-weight-normal">Please sign in</h1>

<?php
    //      Display the error message if the credentials were incorrect
    if ($access == "Denied")
    {
?>
        <div class="alert alert-danger" role="alert">
            Invalid credentials entered. Please try again.
        </div>
<?php
    }
?>

<label for="inputEmail" class="sr-only">Email address</label>
<input type="email" id="inputEmail" name="inputEmail" class="form-control" placeholder="Email
address"

incorrect
    <?php
        //      Re-adds email address as the value in the email address field if the credentials were
        if ($access == "Denied")
        {
            echo " value=\"\$email\"";
        }
    ?> required autofocus>

    <label for="inputPassword" class="sr-only">Password</label>
    <input type="password" id="inputPassword" name="inputPassword" class="form-control"
placeholder="Password" required>

    <button class="btn btn-lg btn-primary btn-block" type="submit">Sign in</button>

</form>

</body>
</html>

```

4.4. Appendix D – Logout.php

```
<?php
//      Create a new session or resume the current one
session_start();

//      Destroy all sessions
session_destroy();

//      Redirect to the welcome page after a 3 second delay
header("refresh:3;url=Welcome.php");
?>

<!doctype html>
<html lang="en">
  <head>

    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
    <title>Rainfall Monitor - Logout</title>

    <!-- Bootstrap core CSS -->
    <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/css/bootstrap.min.css"
integrity="sha384-Vkoo8x4CGsO3+Hhvx8T/Q5PaXtkKtu6ug5TOeNV6gBiFeWPGFN9MuhOf23Q9Ifjh" crossorigin="anonymous">

    <!-- Custom styles for this template -->
    <link href="css/signin.css" rel="stylesheet">

  </head>

  <body class="text-center">

    <div class="col-md-12">
      <h2>You have been logged out.</h2>
      <br>
      <h3>Redirecting to the Welcome page...</h3>
    </div>

  </body>
```

</html>

4.5. Appendix E – TriggerDataImport.php

```
<?php

#####
##
//~~~~~
//      Section for managing unauthenticated users trying to access the page
//~~~~~
~~

//      Create a new session or resume the current one
session_start();

//      Redirect to the welcome page if the 'username' session cookie is blank (i.e. user is not logged in)
if ($_SESSION["username"] == "")
{
    header("Location: Welcome.php");
}

#####
##

?>

<!doctype html>
<html lang="en">
<head>

    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
    <title>Rainfall Monitor - Data Import</title>

    <!-- Bootstrap core CSS -->
    <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/css/bootstrap.min.css"
integrity="sha384-Vkoo8x4CGsO3+Hhxv8T/Q5PaXtkKtu6ug5TOeNV6gBiFeWPGFN9MuhOf23Q9Ifjh" crossorigin="anonymous">

    <!-- Template specific CSS -->
```

```

    <link rel="stylesheet" href="https://getbootstrap.com/docs/4.4/examples/dashboard/dashboard.css">

</head>

<body>

    <nav class="navbar navbar-dark fixed-top bg-dark flex-md-nowrap p-0 shadow">
        <a class="navbar-brand col-sm-3 col-md-2 mr-0" href="Welcome.php">Rainfall Monitor</a>

        <ul class="navbar-nav px-3">
            <li class="nav-item text-nowrap">
                <a class="nav-link" href="Logout.php">Sign out</a>
            </li>
        </ul>
    </nav>

    <div class="container-fluid">

        <div class="row">

            <nav class="col-md-2 d-none d-md-block bg-light sidebar">
                <div class="sidebar-sticky">

                    <ul class="nav flex-column">

                        <li class="nav-item">
                            <a class="nav-link" href="Welcome.php">
                                Welcome
                            </a>
                        </li>

                        <li class="nav-item">
                            <a class="nav-link" href="DataAnalysis.php">
                                Rainfall analysis
                            </a>
                        </li>

                        <li class="nav-item">
                            <a class="nav-link active" href="TriggerDataImport.php">

```

```

                                Import data <span class="sr-only">(current)
                                </a>
                            </li>
                        </ul>

                    </div>
                </nav>

                <main role="main" class="col-md-9 ml-sm-auto col-lg-10 px-4">

                    <div class="d-flex justify-content-between flex-wrap flex-md-nowrap align-items-center
pt-3 pb-2 mb-3 border-bottom">
                        <h1 class="h2">Import data</h1>
                    </div>

                    <p>
                        Click the button below to import rainfall data from the rainfall sensors into
the database.
                    </p>

                    <a href="ImportFromFiles.php" class="btn btn-primary">Import data</a>

                </main>

            </div>
        </div>

        <script src="https://code.jquery.com/jquery-3.4.1.slim.min.js" integrity="sha384-
J6qa4849b1E2+poT4WnyKhv5vZF5SrPo0iEjwBvKU7imGFAV0wwjlYyfoRSJoZ+n" crossorigin="anonymous"></script>
        <!--<script>window.jQuery || document.write('<script
src="/docs/4.4/assets/js/vendor/jquery.slim.min.js"></script>')</script>-->
        <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/js/bootstrap.min.js" integrity="sha384-
wfSDF2E50Y2D1uUdj0O3uMBJnjuUD4Ih7YwaYdliqfktj0Uod8GCExl3Og8ifwB6" crossorigin="anonymous"></script>

    </body>
</html>

```

4.6. Appendix F – ImportFromFiles.php

```
<?php

//#####
##
//~~~~~
~~
//      Section for managing unauthenticated users trying to access the page
//~~~~~
~~

//      Create a new session or resume the current one
session_start();

//      Redirect to the log-in page if the 'username' session cookie is blank (i.e. user is not logged in)
if ($_SESSION["username"] == "")
{
    header("Location: Login.php");
}

//#####
##

//#####
##
//~~~~~
~~
//      Function for returning the sensor file list
//~~~~~
~~

function ReturnSensorList()
{
    include "ConnInfo.php";

    $queryDate = "      SELECT

                                s.SensorID
```

```

        ,      s.Filename

        FROM  sensors s";

//      Set variable with the data from the query
$result = mysqli_query($link, $queryDate);

//      Set variable with the row count from the query
$rowCount = mysqli_num_rows($result);

//      Check for rows
if ($rowCount > 0)
{
    //      Iterate through array of query results and display on-screen
    while ($r = mysqli_fetch_assoc($result))
    {
        //      Add the sensor details to the sensorArray array with the sensor ID as the key
        $sensorArray[$r["SensorID"]] = $r["Filename"];
    }

    //      Return the sensorList array
    return $sensorArray;
}
else
{
    return null;
}
}

//#####

##
?>

<!doctype html>
<html lang="en">
<head>

    <meta charset="utf-8">

```

```

<meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
<title>Rainfall Monitor - Data Import</title>

<!-- Bootstrap core CSS -->
<link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/css/bootstrap.min.css"
integrity="sha384-Vkoo8x4CGsO3+Hhxv8T/Q5PaXtkKtu6ug5TOeNV6gBiFeWPGFN9MuhOf23Q9Ifjh" crossorigin="anonymous">

<!-- Template specific CSS -->
<link rel="stylesheet" href="https://getbootstrap.com/docs/4.4/examples/dashboard/dashboard.css">

</head>

<body>

<nav class="navbar navbar-dark fixed-top bg-dark flex-md-nowrap p-0 shadow">
  <a class="navbar-brand col-sm-3 col-md-2 mr-0" href="Welcome.php">Rainfall Monitor</a>

  <ul class="navbar-nav px-3">
    <li class="nav-item text-nowrap">
      <a class="nav-link" href="Logout.php">Sign out</a>
    </li>
  </ul>
</nav>

<div class="container-fluid">

  <div class="row">

    <nav class="col-md-2 d-none d-md-block bg-light sidebar">
      <div class="sidebar-sticky">

        <ul class="nav flex-column">

          <li class="nav-item">
            <a class="nav-link" href="Welcome.php">
              Welcome
            </a>
          </li>
        </ul>
      </div>
    </nav>
  </div>
</div>

```

```

        <li class="nav-item">
            <a class="nav-link" href="DataAnalysis.php">
                Rainfall analysis
            </a>
        </li>

        <li class="nav-item">
            <a class="nav-link" href="TriggerDataImport.php">
                Import data
            </a>
        </li>
    </ul>

</div>
</nav>

<main role="main" class="col-md-9 ml-sm-auto col-lg-10 px-4">

    <div class="d-flex justify-content-between flex-wrap flex-md-nowrap align-items-center
pt-3 pb-2 mb-3 border-bottom">
        <h1 class="h2">Import results</h1>
    </div>

    <p>

    <?php
        //      Include file with connection to the database
        include "ConnInfo.php";

        //      Call function to return the sensor file list
        $sensorFileList = ReturnSensorList();

        //      Check whether $sensorFileList was set (i.e. sensors found)
        if (isset($sensorFileList))
        {
            //      Declare variable used for counting failed row imports
            $failedRowCount = 0;

```

```

in the code

(?, ?, ?)";

imports

not open file.<p>");

the file isn't empty)

filesize($sensorFileList[$key]);

// Declare a variable containing the SQL query that will be used later
$query = "INSERT INTO rainfalllog (Depth, LogDateTime, SensorID) VALUES

// Loop through each element in $sensorFileList
foreach ($sensorFileList as $key => $sensorFileListData)
{
    echo "<p>Beginning import of file $sensorFileList[$key]...<br><br>";

    // Get the sensor ID from the array key
    $sensorNumber = $key;

    // Declare variables for counting total rows and successful

    $rowCount = 0;
    $insertedRowCount = 0;

    // Check whether the file exists
    if (file_exists($sensorFileList[$key]))
    {
        // Open file for reading (or die handles error)
        $fileLink = fopen($sensorFileList[$key], "r") or die("Could

        // Check to see if the file size is greater than 0 (i.e.

        if (filesize($sensorFileList[$key]) > 0)
        {
            // Read all data from the file and close it
            $data = fread($fileLink,

            fclose($fileLink);

            // Declare arrays for values
            $incomingData = array();
            $values = array();

```

split on line breaks

array (i.e. each row in the file)

row

split on spaces

of values in the row. If not, set the result to 0

\$rowValues array. The date and time are concatenated for the INSERT

connection set in the ConnInfo.php file

query and bind the parameters to the statement

\$dateTime, \$sensorNumber);

```
//      Populate $incomingData array with the file data
$incomingData = explode(chr(13), $data);

//      Loop through each element in the $incomingData
foreach ($incomingData as $row)
{
    //      Increment row counter
    $rowCount++;

    //      Remove any whitespace from each end of the
    $row = trim($row);

    //      Populate $rowValues array with the file data
    $rowValues = explode(" ", $row);

    //      Check whether there are the correct number
    if (count($rowValues) == 3)
    {
        //      Set variables with the values from the
        $dateTime = $rowValues[1]." ".$rowValues[2];
        $depth = intval(trim($rowValues[0]));

        //      Initialise a new statement with the
        $stmt = mysqli_stmt_init($link);

        //      Prepare the statement with the INSERT
        mysqli_stmt_prepare($stmt, $query);
        mysqli_stmt_bind_param($stmt, "isi", $depth,
```

result

counter if \$result equals 1

file_exists("failedImports.txt");

append mode

"a");

of data

". \$sensorNumber. "\n";

fopen operation, add the column headings to the start of the file

importTime depth readingDate readingTime sensor\n");

```
//      Execute the statement and return the
$result = mysqli_stmt_execute($stmt);
}
else
{
    $result = 0;
}

//      Check the result of the execution
if ($result == 1)
{
    //      Increment successful row import

    $insertedRowCount++;
}
else
{
    //      Check whether file already exists
    $fileCheck =

    //      Open the failed imports text file in
    $failureFile = fopen("failedImports.txt",

    //      Create the string containing the row
    $rowText = date("Y-m-d H:i:s")." ".$row."

    //      If the file didn't exist before the
    if ($fileCheck == false)
    {
        fwrite($failureFile, "importDate

    }
```

```

the file

informing them of failed import of the row to the database
danger\" role=\"alert\">Error running query on row $rowCount with following values: $depth, $dateTime,
$sensorNumber. </div><br>";

informing them of failed import of the row to the database
danger\" role=\"alert\">Error running query on row $rowCount with following row: $row. </div><br>";

$rowCount rows.</p>";

//      Append the row of data to the end of
fwrite($failureFile, $rowText);
//      Close the file
fclose($failureFile);

if (count($rowValues) == 3)
{
    //      Echo error message to user
    echo "<div class=\"alert alert-
}
else
{
    //      Echo error message to user
    echo "<div class=\"alert alert-
}

//      Increment count of failed rows
$failedRowCount++;
}

echo "Successfully imported $insertedRowCount of

//      Open the file for writing to truncate it
$fileLink = fopen($sensorFileList[$key], "w");

//      Close the file
fclose($fileLink);

//      Check how many rows failed to import. If greater
than 0, display a message to the user informing them they have been written to a separate text file

```

```

                                if ($failedRowCount > 0)
                                {
                                    echo "<div class=\"alert alert-danger\"
role=\"alert\">$failedRowCount rows failed to import into the database. They have been written to the file
<b>failedImports.txt</b>.</div><br>";
                                }
                                }

                                //      Close the file if it was empty and echo out a message to
the user
                                else
                                {
                                    fclose($fileLink);

                                    echo "The file is empty. </p>";
                                }
                                }
                                else
                                {
                                    //      Alert informing using of missing file
                                    echo "<div class=\"alert alert-danger\"
role=\"alert\">$sensorFileList[$key] not present.</div><br>";
                                }
                                }

                                }
                                else
                                {
                                    echo "<p>No sensors found in the database.</p>";
                                }
                                }

                                ?>

                                </p>

                                </main>

                                </div>
</div>

```

```
<script src="https://code.jquery.com/jquery-3.4.1.slim.min.js" integrity="sha384-
J6qa4849blE2+poT4WnyKhv5vZF5SrPo0iEjwBvKU7imGFAV0wwjlYyfoRSJoZ+n" crossorigin="anonymous"></script>
<script src="https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/js/bootstrap.min.js" integrity="sha384-
wfSDF2E50Y2D1uUdj0O3uMBJnjuUD4Ih7YwaYdliqfktj0Uod8GCExl3Og8ifwB6" crossorigin="anonymous"></script>

</body>
</html>
```

4.7. Appendix G – DataAnalysis.php

```
<?php

include "ConnInfo.php";

//#####
##
//~~~~~
~~
//      Section for managing unauthenticated users trying to access the page
//~~~~~
~~

//      Create a new session or resume the current one
session_start();

//      Redirect to the log-in page if the 'username' session cookie is blank (i.e. user is not logged in)
if ($_SESSION["username"] == "")
{
    header("Location: Login.php");
}

//#####
##

//#####
##
//~~~~~
~~
//      Function for returning the first or last log date. The aggregation type (e.g. MIN or MAX) is passed in
as a //      parameter
//~~~~~
~~

function ReturnDate(string $aggType)
{
```

```

include "ConnInfo.php";

$queryDate = "      SELECT
                                $aggType(rf.LogDateTime) AS Result
                                FROM  rainfalllog rf";

//      Set variable with the data from the query
$result = mysqli_query($link, $queryDate);

//      Set variable with the row count from the query
$rowCount = mysqli_num_rows($result);

//      Check for rows
if ($rowCount > 0)
{
    //      Iterate through array of query results
    while ($r = mysqli_fetch_array($result))
    {
        //      If the aggregation type was MAX, add 1 day to the result of the query and convert to a
date,
        //      otherwise just convert the result to a date
        if ($aggType == "MAX")
        {
            $date = date('Y-m-d', strtotime($r["Result"]. ' + 1 days'));
        }
        else
        {
            $date = date('Y-m-d', strtotime($r["Result"]));
        }

        //      Return the date value
        return $date;
    }
}
else
{
    //      Get today's date
    $date = date('Y-m-d', getdate());
}

```

```

        //      Return the date value
        return $date;
    }
}

//#####

##

//#####

##
//~~~~~
~~
//      Function for returning the list of sensors in the data
//~~~~~
~~

function ReturnSensorArray()
{
    include "ConnInfo.php";

    $querySelect = "  SELECT
                                DISTINCT
                                rf.SensorID
                                ,      s.SensorName
                                FROM  rainfalllog rf
                                INNER JOIN  sensors s    ON      rf.SensorID = s.SensorID";

    //      Set variable with the data from the query
    $result = mysqli_query($link, $querySelect);

    //      Set variable with the row count from the query
    $rowCount = mysqli_num_rows($result);

    //      Declare array for list of sensors
    $sensorArray = array();

```

```

//      Check for rows
if ($rowCount > 0)
{
    //      Iterate through array of query results
    while ($r = mysqli_fetch_assoc($result))
    {
        //      Add the sensor details to the sensorArray array with the sensor ID as the key
        $sensorArray[$r["SensorID"]] = $r["SensorName"];
    }

    //      Return the sensorList array
    return $sensorArray;
}
else
{
    return null;
}
}

//#####

##

//#####

##

//~~~~~

~~

//      Function for returning encoded JSON based on the passed in query and date parameters
//~~~~~

~~

function ReturnChartJSON(string $SQLQuery, string $startDate, string $endDate)
{
    include "ConnInfo.php";

    //      Initialise a new statement with the connection set in the ConnInfo.php file
    $stmt = mysqli_stmt_init($link);

```

```

//      Prepare the statement with the query and bind the parameters to the statement
mysqli_stmt_prepare($stmt, $SQLQuery);
mysqli_stmt_bind_param($stmt, "ss", $startDate, $endDate);

//      Execute the statement and return the result
$result = mysqli_stmt_execute($stmt);

//      Set variable with the data from the query
$data = mysqli_stmt_get_result($stmt);

//      Check for rows
if (mysqli_num_rows($data) > 0)
{
    $chartData = array();

    //      Add the chart data to an array
    while ($r = mysqli_fetch_assoc($data))
    {
        $chartData[] = $r;
    }

    //      Encode the array of chart data as JSON
    $chartJSON = json_encode($chartData, JSON_PRETTY_PRINT);

    return $chartJSON;
}
else
{
    //      No rows returned for query
    return "";
}
}

//#####
##

```

```

//      Call function to return an array of sensor details. The result of the function will indicate whether
there //      is any rainfall data in the database and therefore what should be displayed on the page (null indicates
no //      rainfall data
$sensorList = ReturnSensorArray();

//      Check to see whether $snesorList was set (i.e. there is rainfall data in the database)
if (isset($sensorList))
{

//#####
##

//~~~~~
~~

//      Section for setting variables used for later queries based on whether values were POSTed to the
page

//~~~~~
~~

    if (empty($_POST))
    {
        //      Create an array of all the sensor IDs (i.e. all sensors will be selected by default)
        $selectedSensorList = array_keys($sensorList);

        //      Set variable with a comma-separated string of selected sensor IDs
        $selectedSensorIDs = join(",", $selectedSensorList);

        //      Set start and end dates based on the minimum and maximum dates values in the rainfall data
        $startDate = ReturnDate("MIN");
        $endDate = ReturnDate("MAX");
    }
    else
    {
        //      Set the variable with the POSTed values
        $selectedSensorList = $_POST["inputSensor"];
    }

```

```

        $startDate = $_POST["inputStartDate"];
        $endDate = $_POST["inputEndDate"];

        //      Set variable with a comma-separated string of selected sensor IDs
        $selectedSensorIDs = join(",", $selectedSensorList);
    }

    ##

    //#####

    ##

    //#####

    //~~~~~

    ~~

        //      Section for option list for the sensor input on the filter form

    //~~~~~

    ~~

    $optionList = "";

    //      Iterate through array of sensors
    foreach ($sensorList as $key => $sensorListData)
    {
        //      Check if the sensor has been selected. If so, add 'selected' property to the element
        if (in_array($key, $selectedSensorList) === true)
        {
            $option = "<option selected value=\"\".$key.\">\".$sensorList[$key].\"</option>\";
        }
        else
        {
            $option = "<option value=\"\".$key.\">\".$sensorList[$key].\"</option>\";
        }
    }

```

```

        //      Append the element to the string of HTML
        $optionList = $optionList.$option;
    }

    //#####

##

    //#####

##

    //~~~~~

~~
    data    //      Section for querying the database for rainfall data and populating a variable with the returned
           //      formatted into a HTML table ready to be presented later on in the code

    //~~~~~

~~

    $tableContent = "";

    $query = "  SELECT
                rf.LogID
                ,   rf.Depth
                ,   rf.LogDateTime
                ,   s.SensorName

            FROM  rainfalllog rf

            INNER JOIN  sensors s          ON      rf.SensorID = s.sensorID

            WHERE
                s.SensorID IN ($selectedSensorIDs)
            AND
                rf.LogDateTime >= ?

```

```

        AND
            rf.LogDateTime <= ?

    ORDER BY
        rf.LogDateTime DESC";

//    Initialise a new statement with the connection set in the ConnInfo.php file
$stmt = mysqli_stmt_init($link);

//    Prepare the statement with the query and bind the parameters to the statement
mysqli_stmt_prepare($stmt, $query);
mysqli_stmt_bind_param($stmt, "ss", $startDate, $endDate);

//    Execute the statement and return the result
$result = mysqli_stmt_execute($stmt);

//    Set variable with the data from the query
$data = mysqli_stmt_get_result($stmt);

//    Set variable with the row count from the query
$rowCount = mysqli_num_rows($data);

//    Check for rows
if ($rowCount > 0)
{
    //    Iterate through array of query results and display on-screen
    while ($r = mysqli_fetch_array($data))
    {
        //    Create the HTML for a table row using the row data
        $tableRow = "
                                <tr>
                                <td>".$r["LogID"]."</td>
                                <td>".$r["LogDateTime"]."</td>
                                <td>".$r["Depth"]."</td>
                                <td>".$r["SensorName"]."</td>
                                </tr>";

        //    Append the table row string to the table content string
        $tableContent = $tableContent.$tableRow;
    }
}

```

```

    }

    //#####
##
    //~~~~~
~~
    //    Section for querying the database for rainfall data and populating a variable with the
returned data    //    encoded as JSON to be used for the Chart.js

    //~~~~~
~~

    $queryColumns = "";

    //    Loop through each sensor ($key is the sensor ID) and create the relevant aggregate function
for the SELECT    //    statement
    foreach ($sensorList as $key => $sensorListData)
    {
        $queryColumns = $queryColumns.",    SUM(IF(SensorID = $key, Depth, NULL)) AS
'$sensorList[$key]';
    }

    $query = "  SELECT

                        LogDateTime"
                    ".$queryColumns."

                FROM

                    rainfalllog

                WHERE

                    SensorID IN ($selectedSensorIDs)
                AND

                    LogDateTime >= ?
                AND

                    LogDateTime <= ?

```

```

GROUP BY
    LogDateTime;";

//    Call the ReturnChartJSON function to get the JSON encoded result of the above query
$firstChartJSON = ReturnChartJSON($query, $startDate, $endDate);

##

//#####

##

//#####

//~~~~~
~~
//    Section for querying the database for rainfall statistics and populating a variable with the
returned data
//    encoded as JSON to be used for the Chart.js

//~~~~~
~~

$query = "  SELECT
            LogDateTime
            ,    ROUND (AVG (Depth), 1)          AS    'Average Depth'
            ,    ROUND (MAX (Depth), 1)          AS    'Maximum Depth'
            ,    ROUND (MIN (Depth), 1)          AS    'Minimum Depth'

        FROM
            rainfalllog

        WHERE
            SensorID IN ($selectedSensorIDs)
        AND
            LogDateTime >= ?

```

```

        AND
            LogDateTime <= ?

        GROUP BY
            LogDateTime;";

    //    Call the ReturnChartJSON function to get the JSON encoded result of the above query
    $thirdChartJSON = ReturnChartJSON($query, $startDate, $endDate);

    //#####
##
        }
        else
        {
            //    No rows returned for query
            $returnMessageNoData = "No data found matching those details.";
        }

    //#####
##

    }

?>

<!doctype html>
<html lang="en">
    <head>

        <meta charset="utf-8">
        <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
        <title>Rainfall Monitor - Data Analysis</title>

        <!-- Bootstrap core CSS -->

```

```

    <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/css/bootstrap.min.css"
    integrity="sha384-Vkoo8x4CGsO3+Hhxv8T/Q5PaXtkKtu6ug5TOeNV6gBiFeWPGFN9MuhOf23Q9Ifjh" crossorigin="anonymous">

    <!-- Template specific CSS -->
    <link rel="stylesheet" href="https://getbootstrap.com/docs/4.4/examples/dashboard/dashboard.css">

    <!-- Bootstrap table CSS -->
    <link href="https://unpkg.com/bootstrap-table@1.15.5/dist/bootstrap-table.min.css" rel="stylesheet">

</head>

<body>

    <nav class="navbar navbar-dark fixed-top bg-dark flex-md-nowrap p-0 shadow">
        <a class="navbar-brand col-sm-3 col-md-2 mr-0" href="Welcome.php">Rainfall Monitor</a>

        <ul class="navbar-nav px-3">
            <li class="nav-item text-nowrap">
                <a class="nav-link" href="Logout.php">Sign out</a>
            </li>
        </ul>
    </nav>

    <div class="container-fluid">

        <div class="row">

            <nav class="col-md-2 d-none d-md-block bg-light sidebar">
                <div class="sidebar-sticky">

                    <ul class="nav flex-column">

                        <li class="nav-item">
                            <a class="nav-link" href="Welcome.php">
                                Welcome
                            </a>
                        </li>

                        <li class="nav-item">

```

3 mt-4 mb-1 text-muted">

```

        <a class="nav-link active" href="DataAnalysis.php">
            Rainfall analysis <span class="sr-only">(current)
        </a>
    </li>

    <li class="nav-item">
        <a class="nav-link" href="TriggerDataImport.php">
            Import data
        </a>
    </li>

</ul>

<h6 class="sidebar-heading d-flex justify-content-between align-items-center px-
    <span>Preset filters</span>
</h6>

<ul class="nav flex-column mb-2">

    <li class="nav-item">
        <a class="nav-link" href="" onclick="ApplyFilter('Current month');">
            Current month
        </a>
    </li>

    <li class="nav-item">
        <a class="nav-link" href="" onclick="ApplyFilter('Past year');">
            Past year
        </a>
    </li>

    <li class="nav-item">
        <a class="nav-link" href="" onclick="ApplyFilter('Past week');">
            Past week
        </a>
    </li>

</ul>
```

```

        </div>
    </nav>

    <main role="main" class="col-md-9 ml-sm-auto col-lg-10 px-4">

        <div class="d-flex justify-content-between flex-wrap flex-md-nowrap align-items-center
pt-3 pb-2 mb-3 border-bottom">
            <h1 class="h2">Rainfall analysis</h1>
        </div>

        <?php
            //      Handles what is displayed if there is no rainfall data
            if (isset($selectedSensorIDs))
            {
                ?>
                <ul class="nav nav-pills mb-3" id="chartTabs" role="tablist">

                    <li class="nav-item">
                        <a class="nav-link active" id="pills-firstChart-tab" data-toggle="pill"
href="#pills-firstChart" role="tab" aria-controls="pills-firstChart" aria-selected="true">Rainfall over time (line
chart)</a>
                    </li>

                    <li class="nav-item">
                        <a class="nav-link" id="pills-secondChart-tab" data-toggle="pill"
href="#pills-secondChart" role="tab" aria-controls="pills-secondChart" aria-selected="false">Rainfall over time (bar
chart)</a>
                    </li>

                    <li class="nav-item">
                        <a class="nav-link" id="pills-thirdChart-tab" data-toggle="pill"
href="#pills-thirdChart" role="tab" aria-controls="pills-thirdChart" aria-selected="false">Rainfall statistics over
time</a>
                    </li>

                </ul>

                <div class="tab-content" id="pills-tabContent">

```

```

aria-labelledby="pills-home-tab">
    <div class="tab-pane fade show active" id="pills-firstChart" role="tabpanel"

there was
    <?php
        //      Check to see if there was any data returned. Display the chart if

        if (!isset($returnMessageNoData))
        {
    ?>
        <div style="height: 50vh;">
            <canvas id="firstChart"></canvas>
        </div>
    <?php
        }
        else
        {
            //      Display alert if there was no data returned
            echo "<div class=\"alert alert-danger\"
role=\"alert\">\".$returnMessageNoData.</div><br>";
        }
    ?>

</div>

labelledby="pills-secondChart-tab">

there was
    <?php
        //      Check to see if there was any data returned. Display the chart if

        if (!isset($returnMessageNoData))
        {
    ?>
        <div style="height: 50vh;">
            <canvas id="secondChart"></canvas>
        </div>
    <?php
        }

```

```

        else
        {
            //    Display alert if there was no data returned
            echo "<div class=\"alert alert-danger\"
role=\"alert\">\".$returnMessageNoData.</div><br>";
        }
    ?>

</div>

<div class="tab-pane fade" id="pills-thirdChart" role="tabpanel" aria-
labelledby="pills-thirdChart-tab">

    <?php
        //    Check to see if there was any data returned. Display the chart if
there was
        if (!isset($returnMessageNoData))
        {
            ?>
            <div style="height: 50vh;">
                <canvas id="thirdChart"></canvas>
            </div>
            <?php
            }
            else
            {
                //    Display alert if there was no data returned
                echo "<div class=\"alert alert-danger\"
role=\"alert\">\".$returnMessageNoData.</div><br>";
            }
        }
    ?>

</div>

</div>

<h2>Rainfall data</h2>

<form id="filters" action="" method="POST">

```

```

class="form-control" required>
option list

<fieldset>
    <div class="form-row">
        <div class="form-group col-md-4">
            <label for="inputSensor">Sensor</label>
            <select multiple id="inputSensor" name="inputSensor[]"

                <?php
                    //      Echo out the previously generated HTML for the
                        echo $optionList;
                ?>
            </select>
        </div>

        <div class="form-group col-md-4">
            <label for="inputStartDate">Start date</label>
            <input value="<?php
                //      Echo out the previously calculated start date as
                    echo $startDate;
                ?>" type="date" id="inputStartDate" name="inputStartDate"

            </div>

            <div class="form-group col-md-4">
                <label for="inputEndDate">End date</label>
                <input value="<?php
                    //      Echo out the previously calculated end date as the
                        echo $endDate;
                    ?>" type="date" id="inputEndDate" name="inputEndDate"

                </div>
            </div>
        </fieldset>

```

the default value for the input

default value for the input

class="form-control" required>

```

        <div class="btn-group col-md-6 offset-md-3" role="group" aria-label="Filter
buttons">
            <button class="btn btn-primary" type="submit">Filter results</button>
            <a type="button" class="btn btn-outline-primary"
href="DataAnalysis.php">Reset filters</a>
        </div>
    </form>

    <br>

    <div class="table-responsive">

        <table class="table table-striped table-sm" data-toggle="table">
            <thead>
                <tr>
                    <th data-field="logID" data-sortable="true">Log ID</th>
                    <th data-field="dateTime" data-sortable="true">Date and
time</th>
                    <th data-field="depth" data-sortable="true">Depth (mm)</th>
                    <th data-field="sensor" data-sortable="true">Sensor</th>
                </tr>
            </thead>
            <tbody>
                <?php
                    //      Echo out the previously generated HTML for the table rows
                    echo $tableContent;
                ?>
            </tbody>
        </table>

    </div>

    <?php
    }
    else
    {
        //      Display alert if there is no rainfall data in the database
        echo "<div class=\"alert alert-danger\" role=\"alert\">No rainfall data in the
database.</div><br>";
    }

```

```

        ?>

        </main>

    </div>
</div>

<script src="https://code.jquery.com/jquery-3.4.1.min.js" crossorigin="anonymous"></script>
<script src="https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/js/bootstrap.min.js" integrity="sha384-
wfSDF2E50Y2D1uUdj003uMBJnjuUD4Ih7YwaYdliqfktj0Uod8GCExl3Og8ifwB6" crossorigin="anonymous"></script>
<script src="https://cdn.jsdelivr.net/npm/chart.js@2.7.3/dist/chart.min.js"></script>
<script src="js/RainfallMonitor.js"></script>
<script src="https://unpkg.com/bootstrap-table@1.15.5/dist/bootstrap-table.min.js"></script>

<script>

    //      Set variables with the elements the charts will be added to
    var firstChartElement = document.getElementById('firstChart');
    var secondChartElement = document.getElementById('secondChart');
    var thirdChartElement = document.getElementById('thirdChart');

    //      Set variables with the JSON generated by the PHP
    var firstChartJSON = <?php
    echo $firstChartJSON;
    ?>
    var thirdChartJSON = <?php
    echo $thirdChartJSON;
    ?>

    //      Call the CreateLineChart function (see RainfallMonitor.js) for the first chart
    CreateLineChart(firstChartElement, firstChartJSON);

    //      Call the CreateBarChart function (see RainfallMonitor.js) for the second chart
    CreateBarChart(secondChartElement, firstChartJSON);

    //      Call the CreateLineChart function (see RainfallMonitor.js) for the third chart
    CreateLineChart(thirdChartElement, thirdChartJSON);

```

```
</script>
```

```
</body>
```

```
</html>
```

4.8. Appendix H – CheckCredentials.php

```
<?php

//      Create a new session or resume the current one
session_start();

include "ConnInfo.php";

//      Set variables withn the data from the POST
$email = $_POST["inputEmail"];
$password = $_POST["inputPassword"];

$query = "  SELECT
                Username
            FROM  users
            GROUP BY
                Username
                ,      Password
            HAVING
                Username = ?
            AND
                Password = ?";

//      Initialise a new statement with the connection set in the ConnInfo.php file
$stmt = mysqli_stmt_init($link);

//      Prepare the statement with the query and bind the parameters to the statement
mysqli_stmt_prepare($stmt, $query);
mysqli_stmt_bind_param($stmt, "ss", $email, $password);

//      Execute the statement and return the result
$result = mysqli_stmt_execute($stmt);

//      Set variable with the data from the query
$data = mysqli_stmt_get_result($stmt);
```

```

//      Set variable with the row count from the query
$rowCount = mysqli_num_rows($data);

//      Check to see if rows were returned. If so, set session with email address, otherwise POST back to
//      the page that access was denied
if ($rowCount == 1)
{
    $_SESSION["username"] = $email;
    header("Location: Welcome.php");
}
else
{
    ?>
        <form name="form" action="Login.php" method="POST">
            <input type="hidden" name="access" value="Denied">
            <input type="hidden" name="email" value="<?php
                echo $email;
            ?>">
        </form>

        <script type="text/javascript">
            document.form.submit();
        </script>
    <?php
    }
    ?>

```

4.9. Appendix I – RainfallMonitor.js

```
//      Function for generating a random colour

function DynamicColour()
{

    var r = Math.floor(Math.random() * 255);
    var g = Math.floor(Math.random() * 255);
    var b = Math.floor(Math.random() * 255);

    return "rgba(" + r + "," + g + "," + b + ", 0.75)";

};


//      Function for outputting a chart to the defined element

function CreateChart(chartElement, chartJSON)
{

    //      Create a blank array for the chart datasets
    var datasets = [];

    //      Initialise a variabel for counting
    var count = 0;

    //      Iterate through the JSON data. 'key' is the data name
    for (var key in chartJSON[0])
    {
        if (key == "LogDateTime")
        {
            var labels = chartJSON.map(function(s) {
                return s[key];
            });
        }
        else
        {

```

```

        var data = chartJSON.map(function(s) {
            return s[key];
        });

        var lineColor = DynamicColour();

        datasets[count] = {
            label: key,
            data: data,
            borderWidth: 2,
            fill: false,
            backgroundColor: lineColor,
            borderColor: lineColor,
            lineTension: 0
        }

        count++;
    }
}

var myChart = new Chart(chartElement, {
    type: 'line',
    data: {
        labels: labels,
        datasets: datasets
    },
    options: {
        responsive: true,
        maintainAspectRatio: false,
        aspectRatio: 2,
        title: {
            display: true,
            text: 'Rainfall depth over time'
        },
        scales: {
            xAxes: [{
                display: true,
                scaleLabel: {
                    display: true,

```

```

                labelString: 'Date & time'
            },
            ticks: {
                maxRotation: 90
            }
        }],
        yAxes: [{
            scaleLabel: {
                display: true,
                labelString: 'Depth (mm)'
            },
            ticks: {
                beginAtZero: false
            }
        }]
    }
}

});

};

//    Function for applying a filter to the 'filters' form

function ApplyFilter(filter)
{
    //    Prevent the default action of the event
    event.preventDefault();

    //    Set variable with current date
    const currentDate = new Date();

    //    Set variables for the offsets that will be deducted from the start and end dates
    var startDateOffset = 0;
    var endDateOffset = 0;

    //    Switch statement that sets the offsets based on the filter selected
    switch (filter)

```

```

{
    case "Current month":
        startDateOffset = currentDate.getDate() - 1;
        endDateOffset = 0;
        break;
    case "Past year":
        startDateOffset = 365;
        endDateOffset = 0;
        break;
    case "Past week":
        startDateOffset = 7;
        endDateOffset = 0;
        break;
    default:
        break;
}

// Set variable with the start date (current date minus start date offset)
var startDate = new Date(currentDate.getFullYear(), currentDate.getMonth(), (currentDate.getDate() -
startDateOffset));

// Get the start month and day and convert to string
var startMonth = startDate.getMonth() + 1;
var startDay = startDate.getDate();
var startMonthString = String(startMonth);
var startDayString = String(startDay);

// If the start month is less than 10, add the leading 0
if (startMonth < 10)
{
    startMonthString = "0" + String(startMonth);
}

// If the start day is less than 10, add the leading 0
if (startDay < 10)
{
    startDayString = "0" + String(startDay);
}

```

```

        //      Set variable with the end date (current date minus end date offset)
        var endDate = new Date(currentDate.getFullYear(), currentDate.getMonth(), (currentDate.getDate() -
endDateOffset));

        //      Get the end month and day and convert to string
        var endMonth = endDate.getMonth() + 1;
        var endDay = endDate.getDate();
        var endMonthString = String(endMonth);
        var endDayString = String(endDay);

        //      If the end month is less than 10, add the leading 0
        if (endMonth < 10)
        {
            endMonthString = "0" + String(endMonth);
        }

        //      If the end day is less than 10, add the leading 0
        if (endDay < 10)
        {
            endDayString = "0" + String(endDay);
        }

        //      Construct the date strings from the variables
        var startDateString = startDate.getFullYear() + "-" + startMonthString + "-" + startDayString;
        var endDateString = endDate.getFullYear() + "-" + endMonthString + "-" + endDayString;

        //      Set the input fields with the date strings
        document.getElementById("inputStartDate").value = startDateString;
        document.getElementById("inputEndDate").value = endDateString;

        //      Submit the form
        document.getElementById("filters").submit();

    }
    //      Function for generating a random colour

function DynamicColour()
{

```

```

    var r = Math.floor(Math.random() * 255);
    var g = Math.floor(Math.random() * 255);
    var b = Math.floor(Math.random() * 255);

    return "rgba(" + r + "," + g + "," + b + ", 0.75)";

};

//      Function for outputting a line chart to the defined element
function CreateLineChart(chartElement, chartJSON)
{
    //      Create a blank array for the chart datasets
    var datasets = [];

    //      Initialise a variable for counting
    var count = 0;

    //      Iterate through the JSON data. 'key' is the data name
    for (var key in chartJSON[0])
    {
        if (key == "LogDateTime")
        {
            var labels = chartJSON.map(function(s) {
                return s[key];
            });
        }
        else
        {
            var data = chartJSON.map(function(s) {
                return s[key];
            });

            var lineColor = DynamicColour();

            datasets[count] = {
                label: key,

```

```

        data: data,
        borderWidth: 2,
        fill: false,
        backgroundColor: lineColor,
        borderColor: lineColor,
        lineTension: 0
    }

    count++;
}

// Create the chart object
var myChart = new Chart(chartElement, {
    type: 'line',
    data: {
        labels: labels,
        datasets: datasets
    },
    options: {
        responsive: true,
        maintainAspectRatio: false,
        aspectRatio: 2,
        title: {
            display: true,
            text: 'Rainfall depth over time'
        },
        scales: {
            xAxes: [{
                display: true,
                scaleLabel: {
                    display: true,
                    labelString: 'Date & time'
                },
                ticks: {
                    maxRotation: 90
                }
            }],
            yAxes: [{

```

```

        scaleLabel: {
            display: true,
            labelString: 'Depth (mm)'
        },
        ticks: {
            beginAtZero: false
        }
    }
}

});

};

//      Function for outputting a stacked bar chart to the defined element
function CreateBarChart(chartElement, chartJSON)
{
    //      Create a blank array for the chart datasets
    var datasets = [];

    //      Initialise a variabel for counting
    var count = 0;

    //      Iterate through the JSON data. 'key' is the data name
    for (var key in chartJSON[0])
    {
        if (key == "LogDateTime")
        {
            var labels = chartJSON.map(function(s) {
                return s[key];
            });
        }
        else
        {
            var data = chartJSON.map(function(s) {
                return s[key];
            });
        }
    }
}

```

```

    });

    var lineColor = DynamicColour();

    datasets[count] = {
        label: key,
        data: data,
        borderWidth: 2,
        fill: false,
        backgroundColor: lineColor,
        borderColor: lineColor,
        lineTension: 0
    }

    count++;
}

// Create the chart object
var myChart = new Chart(chartElement, {
    type: 'bar',
    data: {
        labels: labels,
        datasets: datasets
    },
    options: {
        responsive: true,
        maintainAspectRatio: false,
        aspectRatio: 2,
        title: {
            display: true,
            text: 'Rainfall depth over time'
        },
        scales: {
            xAxes: [{
                stacked: true,
                display: true,
                scaleLabel: {
                    display: true,

```

```

        labelString: 'Date & time'
    },
    ticks: {
        maxRotation: 90
    }
}],
yAxes: [{
    stacked: true,
    scaleLabel: {
        display: true,
        labelString: 'Depth (mm)'
    },
    ticks: {
        beginAtZero: false
    }
}]
}
}
});

};

//      Function for outputting a pie chart to the defined element
function CreatePieChart(chartElement, chartJSON)
{
    //      Create a blank array for the chart datasets
    var datasets = [];

    //      Initialise a variabel for counting
    var count = 0;

    //      Iterate through the JSON data. 'key' is the data name
    for (var key in chartJSON[0])
    {
        if (key == "SensorName")
        {

```

```

        var labels = chartJSON.map(function(s) {
            return s[key];
        });
    }
    else
    {
        var data = chartJSON.map(function(s) {
            return s[key];
        });

        var lineColor = DynamicColour();

        var bgColours = [];

        for (var value in data)
        {
            bgColours[value] = DynamicColour();
        }

        datasets[count] = {
            label: key,
            data: data,
            borderWidth: 2,
            backgroundColor: bgColours
        }

        count++;
    }
}

// Create the chart object
var myChart = new Chart(chartElement, {
    type: 'pie',
    data: {
        labels: labels,
        datasets: datasets
    },
    options: {
        responsive: true,

```

```

        maintainAspectRatio: false,
        aspectRatio: 2,
        title: {
            display: true,
            text: 'Comparative total rainfall depths (mm) for last day of readings'
        }
    }
});

};

//      Function for applying a filter to the 'filters' form
function ApplyFilter(filter)
{
    //      Prevent the default action of the event
    event.preventDefault();

    //      Set variable with current date
    const currentDate = new Date();

    //      Set variables for the offsets that will be deducted from the start and end dates
    var startDateOffset = 0;
    var endDateOffset = 0;

    //      Switch statement that sets the offsets based on the filter selected
    switch (filter)
    {
        case "Current month":
            startDateOffset = currentDate.getDate() - 1;
            endDateOffset = 0;
            break;
        case "Past year":
            startDateOffset = 365;
            endDateOffset = 0;
            break;
        case "Past week":

```

```

        startDateOffset = 7;
        endDateOffset = 0;
        break;
    default:
        break;
}

//    Set variable with the start date (current date minus start date offset)
var startDate = new Date(currentDate.getFullYear(), currentDate.getMonth(), (currentDate.getDate() -
startDateOffset));

//    Get the start month and day and convert to string
var startMonth = startDate.getMonth() + 1;
var startDay = startDate.getDate();
var startMonthString = String(startMonth);
var startDayString = String(startDay);

//    If the start month is less than 10, add the leading 0
if (startMonth < 10)
{
    startMonthString = "0" + String(startMonth);
}

//    If the start day is less than 10, add the leading 0
if (startDay < 10)
{
    startDayString = "0" + String(startDay);
}

//    Set variable with the end date (current date minus end date offset)
var endDate = new Date(currentDate.getFullYear(), currentDate.getMonth(), (currentDate.getDate() -
endDateOffset));

//    Get the end month and day and convert to string
var endMonth = endDate.getMonth() + 1;
var endDay = endDate.getDate();
var endMonthString = String(endMonth);
var endDayString = String(endDay);

```

```

//      If the end month is less than 10, add the leading 0
if (endMonth < 10)
{
    endMonthString = "0" + String(endMonth);
}

//      If the end day is less than 10, add the leading 0
if (endDay < 10)
{
    endDayString = "0" + String(endDay);
}

//      Construct the date strings from the variables
var startDateString = startDate.getFullYear() + "-" + startMonthString + "-" + startDayString;
var endDateString = endDate.getFullYear() + "-" + endMonthString + "-" + endDayString;

//      Set the input fields with the date strings
document.getElementById("inputStartDate").value = startDateString;
document.getElementById("inputEndDate").value = endDateString;

//      Submit the form
document.getElementById("filters").submit();
}

```