SEWANALYST
USER MANUAL

Josh Burns

Kirsty Watt

Jasmine Neild

Louisa Murray

Marguerite Julia

Adam Davies

TABLE OF CONTENTS

1 - INTRODUCTION

Welcome to the **SewAnalyst** User Manual. For non-technical users, please refer to sections **2 – Website Usage** and **3 - Software Usage** as well as ensuring that you check **1.1.1 - System Requirements** stated below. For developers concerned with maintenance and possible future features, please refer to sections **4 - Setup and Hosting, 5 - Website Implementation Details, 6 - Application Implementation Details** and **7 - Maintenance.**

Access to all code is available at: https://github.com/jbazel/SewAnalyst
(The main branch contains the most up-to-date version of the software)

1.1 - SOFTWARE

The primary function of the SewAnalyst software is to allow the input of data from sewage treatment works; processing it to ultimately produce a PDF report detailing the results with associated figures to visualise the data.

The metrics that the program analyses are:

1. **Population Equivalent (PE)**

   Population Equivalent is a statistic that describes the specific load of a wastewater treatment plant. This means in practice that one person resident in an average house is expected to produce 200 litres of sewage flow containing 60g of biological oxygen demand (**BOD**) per day, which is equivalent to one PE.

2. **Dry Weather Flow (DWF)**

   Dry Weather Flow is defined as the average daily flow to a wastewater treatment works during a period without rainfall. We have opted to calculate DWF as described, as opposed to nonparametric 80% exceeded flow:

DWF = PG + IDWF + E
DWF = Total dry weather flow *(l/d)*
P = Catchment population *(number)*
G = Per capita domestic flow *(l/hd/d)*
IDWF = Dry weather infiltration *(l/d)*
E = Trade effluent flow *(l/d)*

3. Flow to Full Treatment (FFT)

Flow to Full Treatment is a measure of the maximum flow a wastewater treatment plant is able to treat at any given time. These STWs are designed to treat a volume equal to three times the volume of DWF. Therefore, in order to calculate this value, we simply multiply DWF by three.

After data entry and analysis surrounding the three described metrics have been completed, the report details the results in simple English so that the explanations can be understood by users and those outside of the context of sewage analysis. The PDF report is saved onto the user's PC/laptop.

1.1.1 - SYSTEM REQUIREMENTS:

In order to run the SewAnalyst software, your PC or Laptop must meet these minimum requirements:

<table>
<tr><td>Operating System:</td><td colspan="2">MAC OS Ventura 1.0. + (arm architecture)
Windows 7 (onwards)</td></tr>
<tr><td>Minimum Storage Capacity:</td><td colspan="2">100mb
(Note: This is larger than the main program executable, however this is the minimum safe amount accounting for the additional storage needed for resources and generated reports)</td></tr>
<tr><td rowspan="2">RAM (memory):</td><td>Minimum</td><td>Recommended</td></tr>
<tr><td>150mb</td><td>1gb +

(1gb+ is recommended in order to maintain the functionality of your web-browser without causing performance issues)</td></tr>
<tr><td>Web Browser:</td><td colspan="2">Any modern web browser

(Testing of SewAnalyst was performed on Google Chrome)</td></tr>
</table>

1.2 - WEBSITE

In order to make our software accessible and distributable, there has a been a website developed to function as a distribution platform in which **SewAnalyst** may be downloaded. The website also functions as a method for users to upload reports that they have generated and chosen to upload for others to examine and report if they find any erroneous outcomes generated.

1.3 - JUSTIFICATION

The initial requirements for SewAnalyst were outlined by the client, Windrush Against Sewage Pollution (WASP); an organisation committed to protecting the United Kingdom's rivers and ecosystems from the harm caused by illegal sewage dumping and pollution. Currently, there is a lack of tools to aid analysis of STW data for activist groups and companies such as WASP, so their endeavours take time and are often left unconducted. By enabling accurate analysis of reported dumping from these companies by activists around the UK, we endeavour to increase awareness surrounding this issue and hold offending companies accountable. The following is an outline of the initial requirements from our brief and an explanation and justification of the software's solution.

SewAnalyst permits data pertaining to PE, DWF and FFT to be gathered and analysed to answer three questions, and after several meetings and communication with WASP, agreed solutions to their amended requirements are outlined below. As the requirements dictate that the software must determine the answer to three key questions, our software generates a pdf report with the answers to each of the questions respectively separated into their own sections.

Presented in the following table are how we constructed the answers generated onto the report.

1.	**Do water companies accurately and consistently assess PE for individual STW's?**
In order to effectively answer this question, our solution was to allow the user to enter the company's assessment of PE in our software and compare this with SOLAR PE forecast. Each data set is specific to each STW, and so the 'individual' aspect of this question is covered. We calculate and determine significant deviations between this forecasted and reported data and display the results in a simple line graph with an accompanying paragraph of explanation. Accuracy and consistency are evaluated by plotting both the company's recorded PE and the SOLAR PE forecast, so that the difference is visualised.	
2.	**Do water companies accurately calculate DWF and FFT using the prescribed EA guidance?**
Given that this question is a similar framework to the previous question but with different metrics, the program achieves the answer to this question by recalculating and comparing based upon the previous data. A similar line plot and explanatory paragraph are included in order to make the results accessible to non-technical users.	
3.	**Does the EA routinely and robustly Qualitatively Assure (QA) these calculations and measures by the water companies thus ensuring provision of adequate sewage treatment capacity at water company STW's?**
Based on the first two questions, our software assesses whether these deviations from provisioned guidance are significant enough to warrant a conclusion that adequate and qualitatively assured checks are not being carried out.	

2 - WEBSITE USAGE

When the website loads, the user will be presented with the home page of our website:

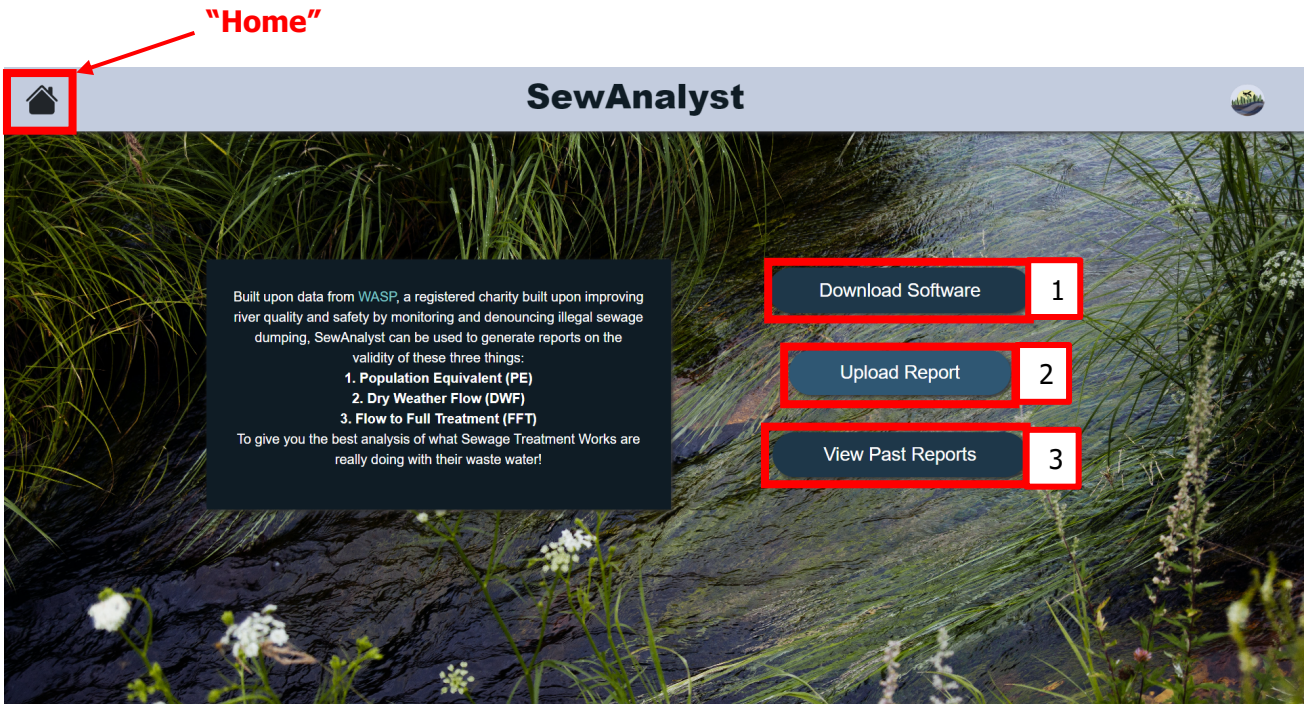
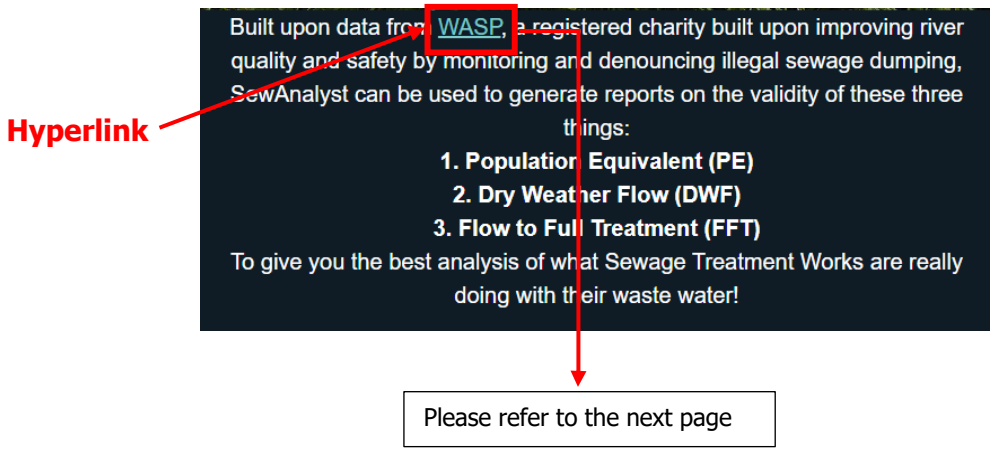
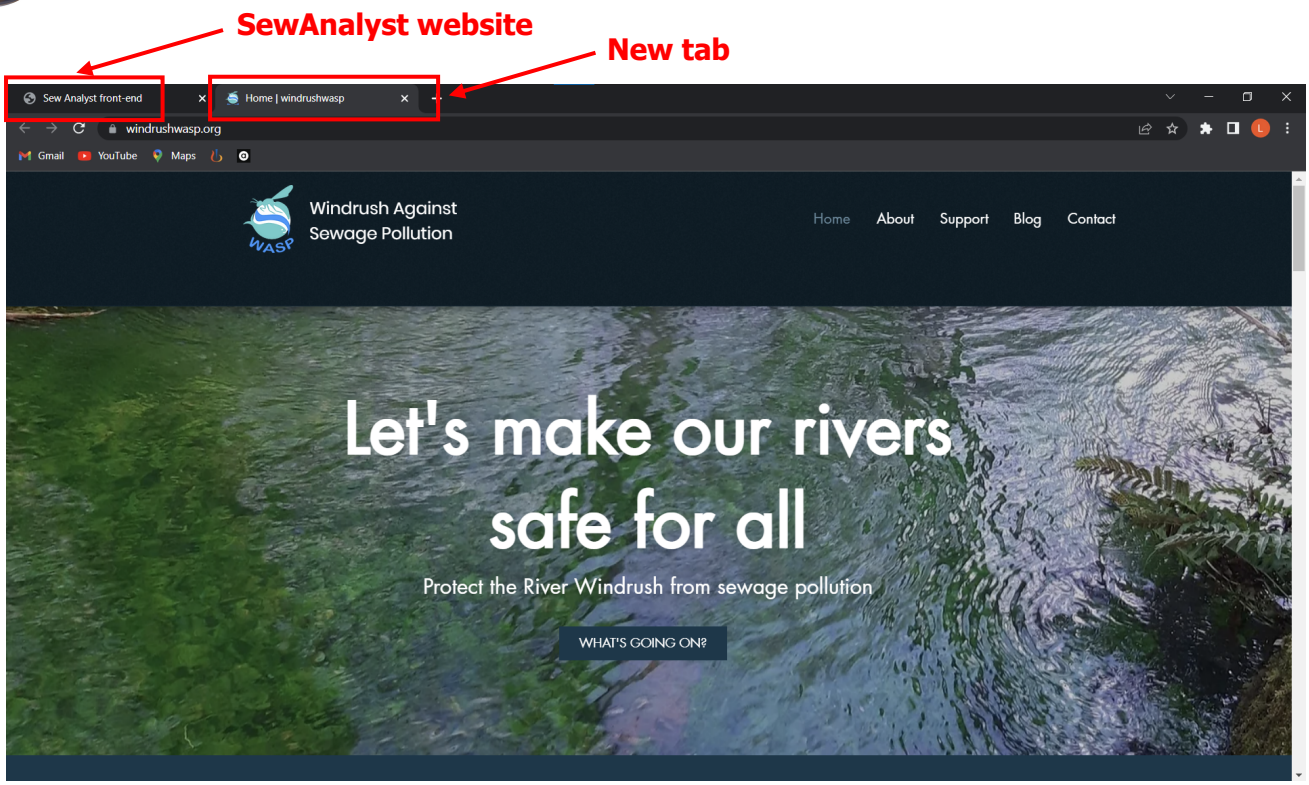


Image of Home Page

The home page contains a short description of what the software aims to do, using the questions given to us on the brief.

This description also contains a hyperlink on the word "WASP". This hyperlink opens a new browser tab when clicked, which will direct the user to the Windrush Against Sewage Pollution website (http://www.windrushwasp.org) so that they can learn more about our WASP.



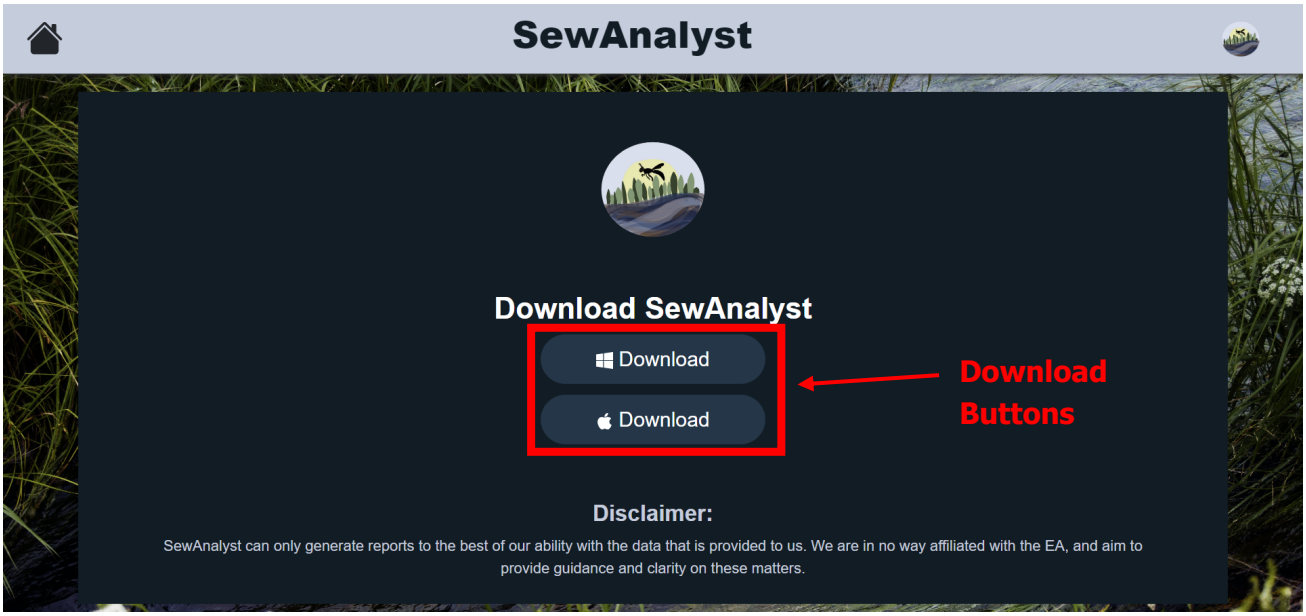


Windrush Again Sewage Pollution's Website (www.windrushwasp.org)

As WASP's website has been opened in a new browser tab, to go back to the SewAnalyst website, the user can simply delete this browser tab or navigate back to their previously opened SewAnalyst website tab. While on the SewAnalyst main website, to return home, press the "home" button located in the top left corner.

2.1 - DOWNLOADING THE SOFTWARE

From the home page of the website, the user should click the "Download Software" button (button 1 on the home page) to get to the Download Software page, where the SewAnalyst software will be available to download.



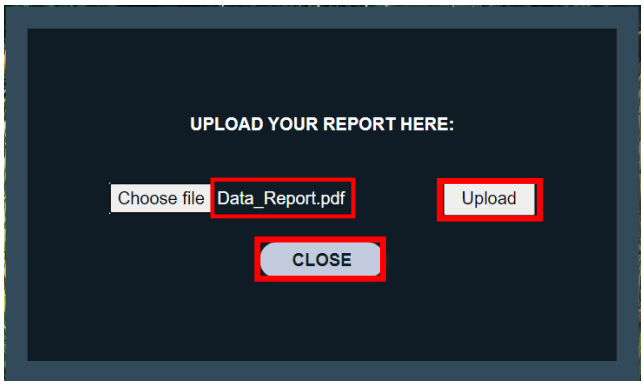
On this page, the user is presented with two "Download" buttons: one for downloading the software for Windows, and one for MAC. The user must select the version of the software corresponding to their operating system.

Below the two "Download" buttons, there is a disclaimer briefly outlining the capabilities of our software and clarifying our lack of affiliation with the EA (Environment Agency), where once pressed, will prompt the respective download for either **SewAnalyst-win.zip** or **SewAnalyst-mac.zip** via the web browser.

2.2 - UPLOADING A REPORT

To upload a report, the user needs to press the "Upload Report" button (button 2 on the home page) on the home page, where they will be presented with the following page:

From here, the user should press the "Choose file" button in order to select a file from their device to upload to the website. If their desired file is a report generated from the software, it will be stored in the 'reports' folder. Once a file has been chosen, the name of the file will appear on the screen as shown below:



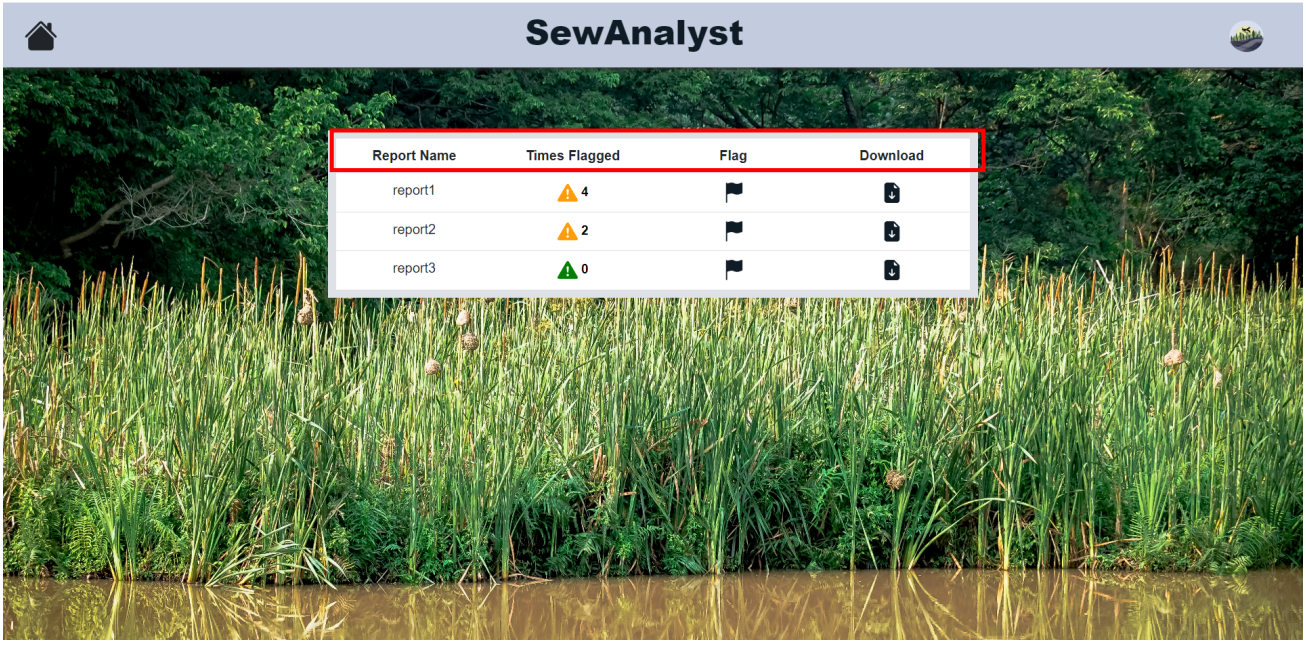
The user should then press the upload button if they are happy to upload the selected report.

Once the user has pressed the upload button, the report will be uploaded to the website and the user will be redirected back to the home page.

If the user decides that they do not wish to upload the report anymore, they can either press the "CLOSE" button or the "Home" button on the top left to be redirected back to the home page.

2.3 - VIEWING PAST REPORTS

To view previously generated reports that have been uploaded to the website, the user should press the "View Past Reports" button (button 3 on the home page) on the home page to be navigated to the view reports page.



This page displays information about past reports in a tabular format.

There are four columns of the table: Report Name, Times Flagged, Flag, and Download.

Each row in the table corresponds to one individual report.

For example, this row in the table contains all the information for one report:

Report Name 1

The Report Name is the unique identifier for that specific report – no two reports will have the same name.

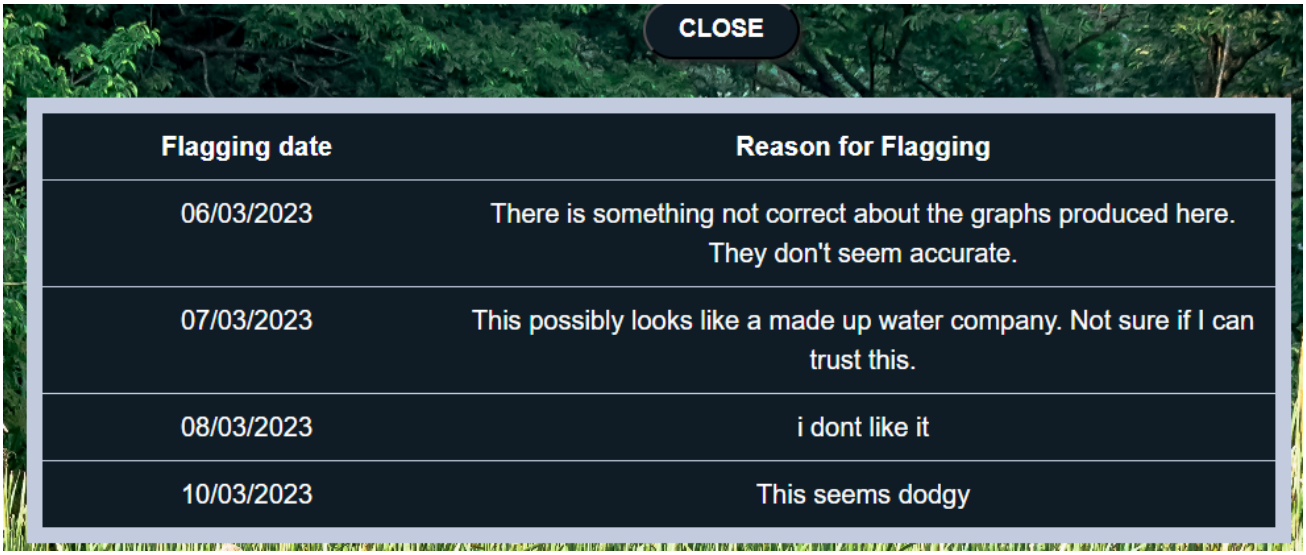
Times Flagged 2

In the Times Flagged column, a warning symbol will appear, and is report dependent (i.e. the warning symbol changes colour depending on the number of times its specific report has been flagged). The thresholds for the three warning symbol "states" are outlined below:

	A **green** triangle will appear if a report has been flagged 0 times.
	An **orange** triangle will appear if a report has been flagged between 1 and 5 times.
	A **red** triangle will appear if a report has been flagged 6 or more times.

Next to the warning symbol is a number, corresponding to the number of times the report has been flagged.

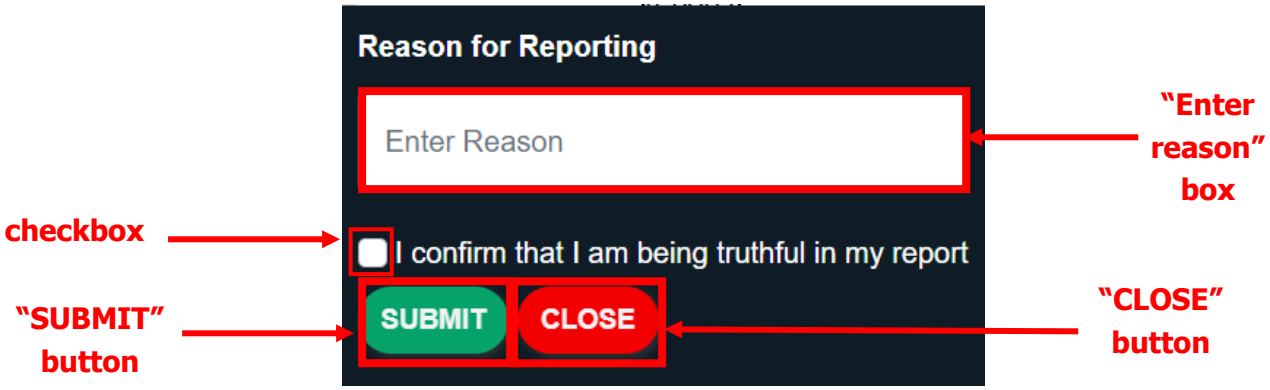
The user may want to read the reasons why a certain report has been flagged. They can do this by clicking the triangle/number in "Times Flagged" column. Once clicked, a table is displayed to the user showing all the reasons the report has been flagged, along with the corresponding dates.



To close this table, the user should press the "CLOSE" button above the table.

Flag Report 

If the user might have a reasonable belief that a certain report is flawed or inaccurate they may click the Flag icon in the "Flag" column and fill out the pop-up to flag this suspicious report.



From the popup displayed, the user must meet two requirements for the form to be able to be submitted:

1) They must fill out a reason for flagging the report.
2) They must check the checkbox to signify that they are being truthful.

Once these 2 requirements have been met, the user must press the "SUBMIT" button to submit their reason for flagging report. Once they do this, the user will be redirected to the home page.

If the user changes their mind and decides that they do not want to flag the report, they can press the "CLOSE" button to return to the rest of the page.

Download Report 4

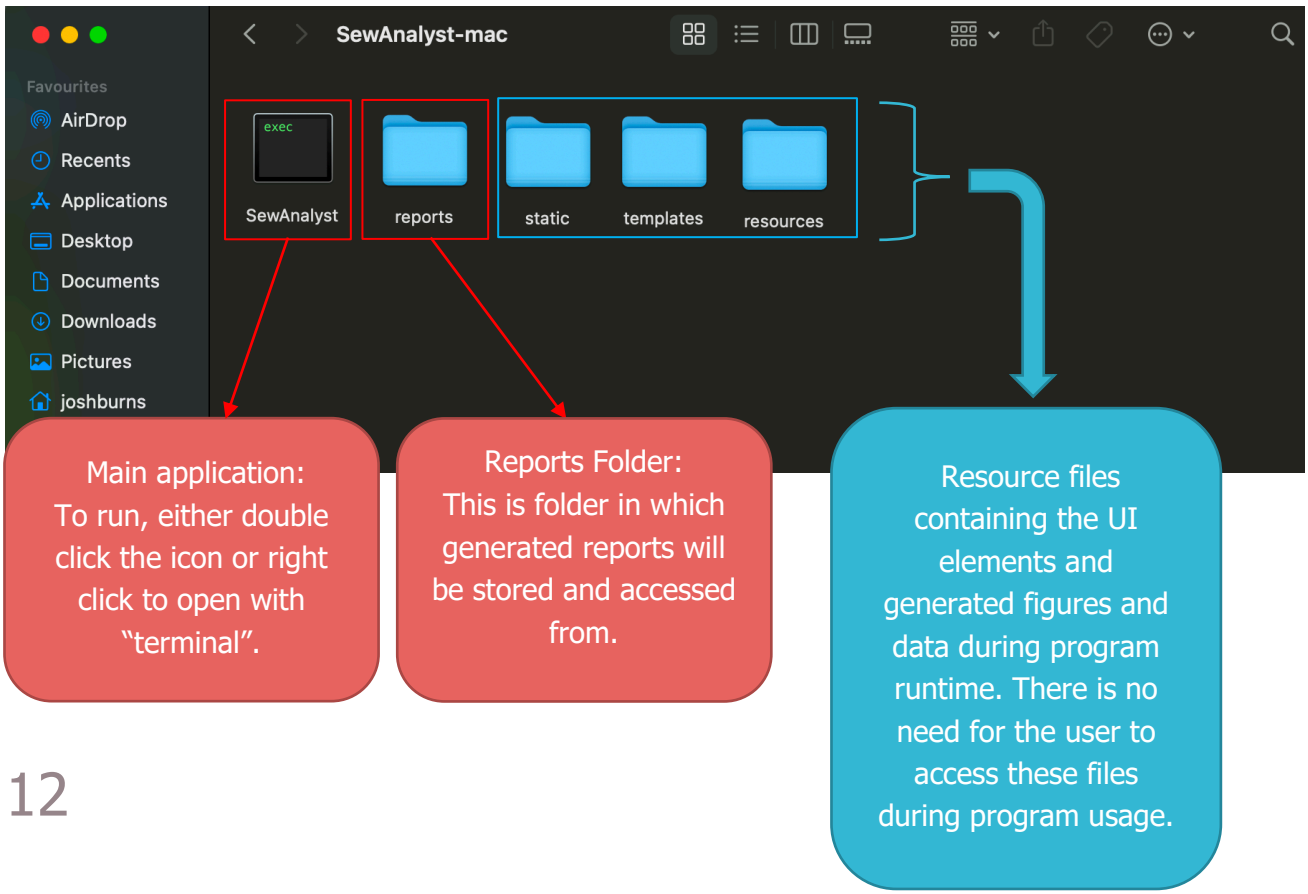
In the Download Report column, the user can click the file icon to download the specific report that corresponds to the report name in the Report Name column. This will download the report onto their device, which they will be able to view in the downloads folder of their computer.

3 - SOFTWARE USAGE

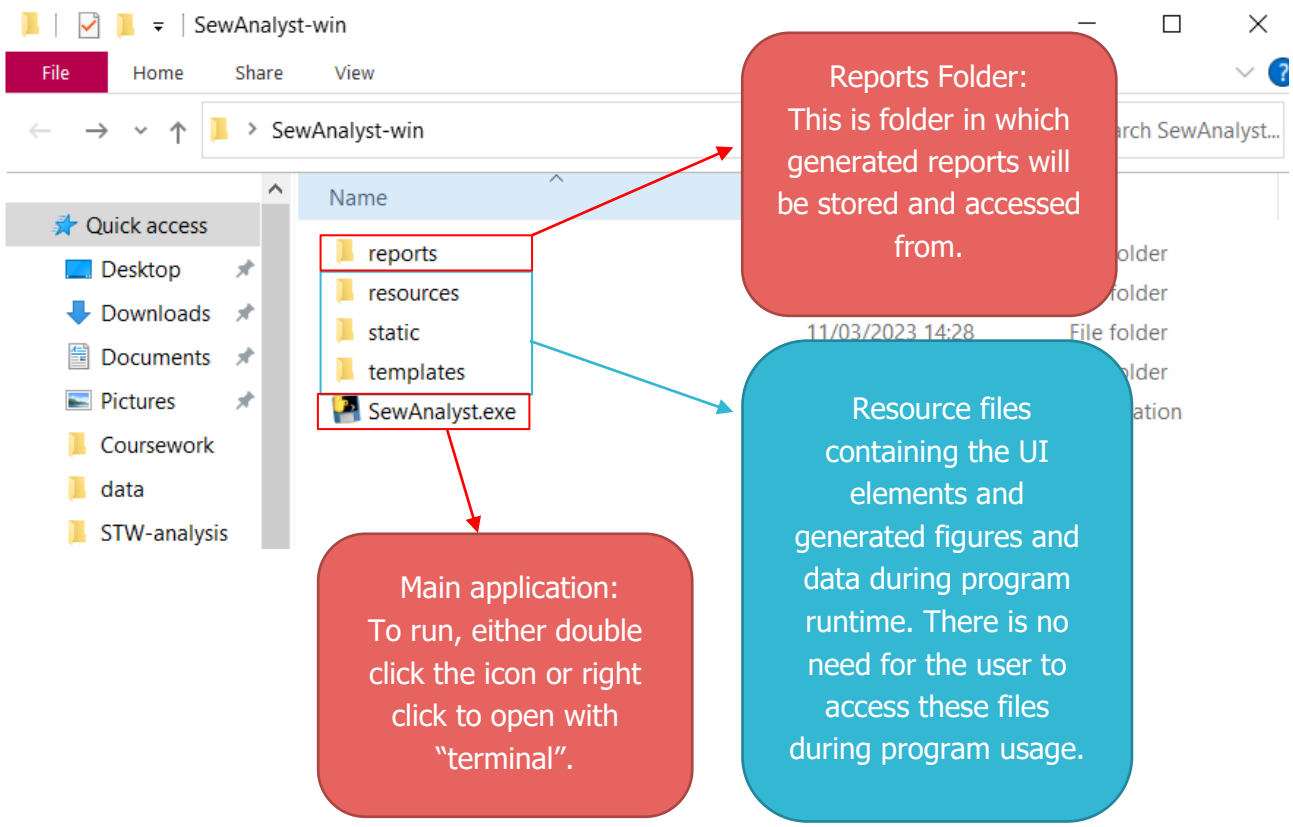
3.1 - OPENING THE SOFTWARE

Once the software has been downloaded, the compressed folder must be unzipped by double-clicking it. It will be found in the default downloads folder of the web browser in which the program is ran. Once this is completed the downloaded file will look like this (*distribution dependent*):

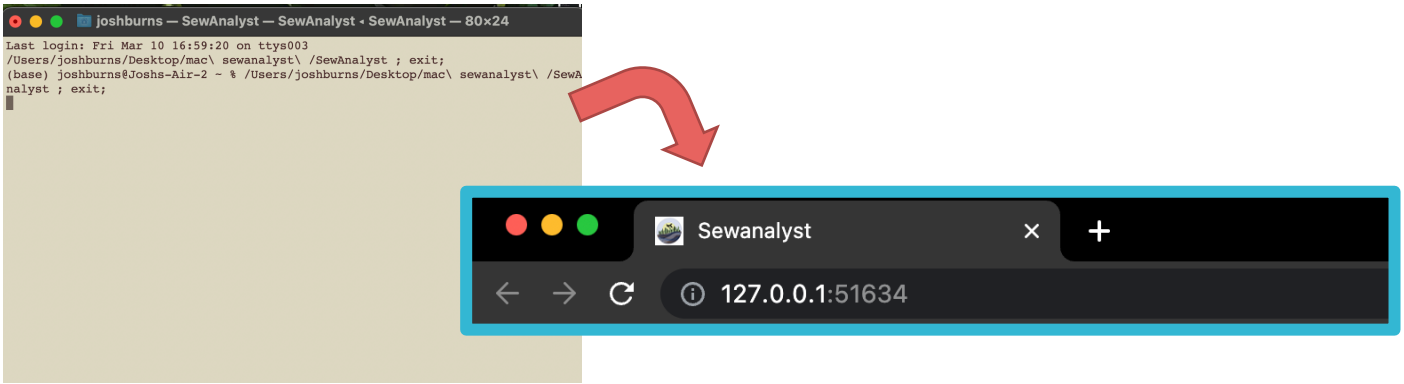
3.1.1 - MAC OS



3.1.2 - WINDOWS:



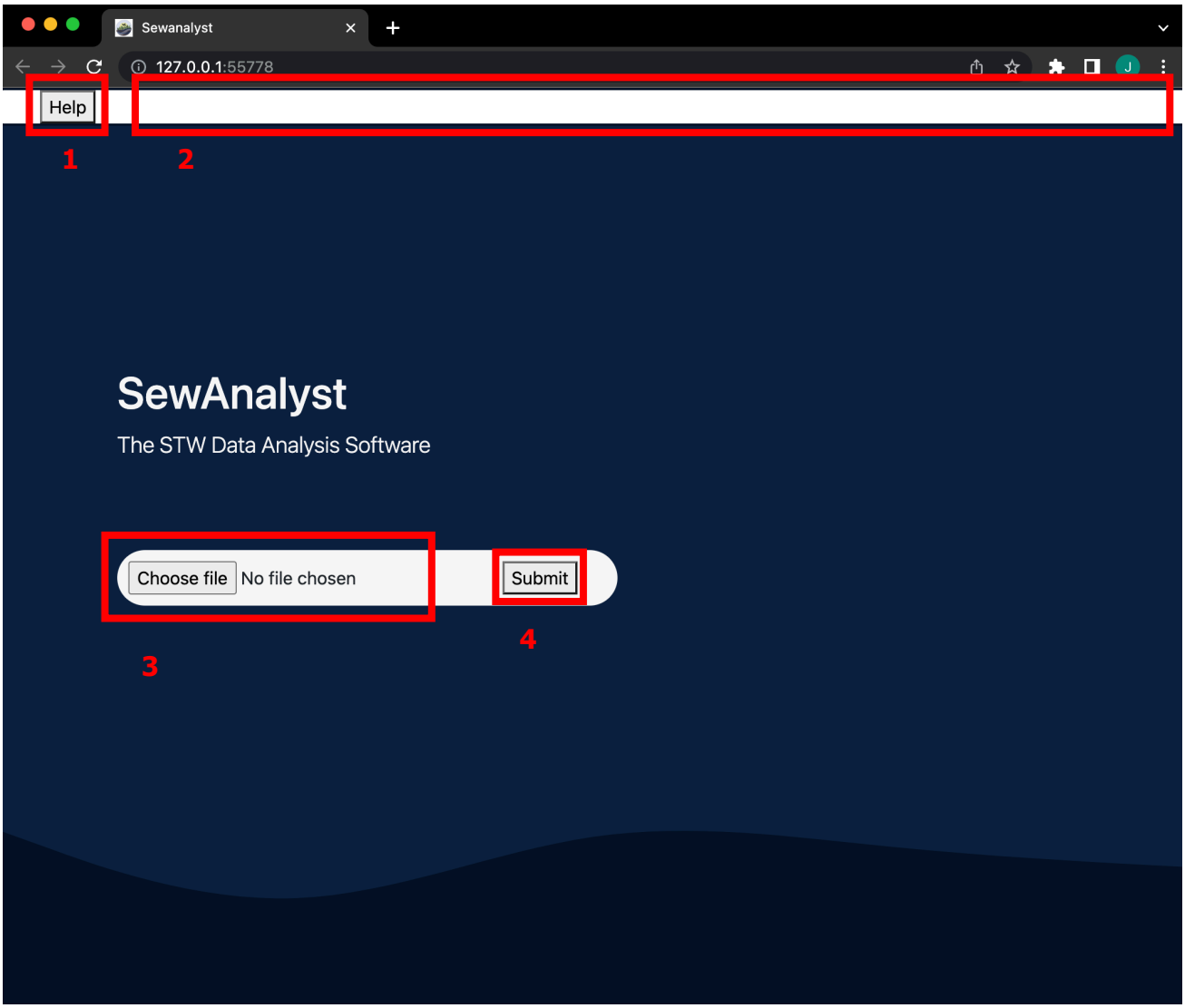
Once the application is launched, the console will open showing an indication of the software launch. After a moment, the software will then launch in the browser.



Please note:

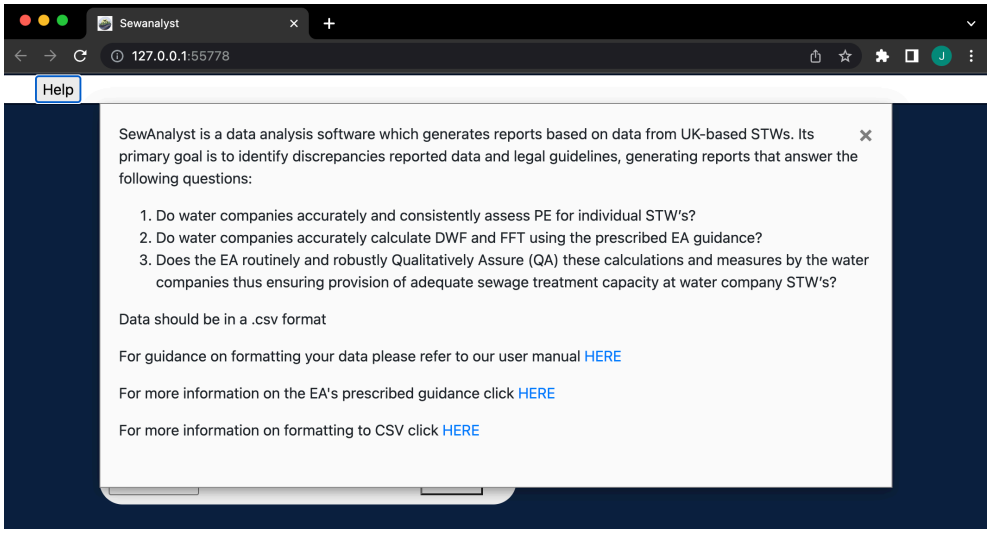
Due to the nature of executable-packaged python the application may take longer than expected to open. Please refer to the console during this process; if no error is given, the application is currently opening.

3.2 - UI GUIDE

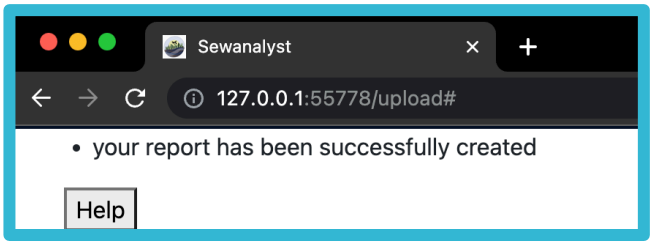


1) Help Button

The help button opens a help prompt that allows the provides basic guidance on software usage as well as links to **EA** guidance, a guide for **CSV Formatting** and a link to the **User Manual** if more detailed guidance is required.



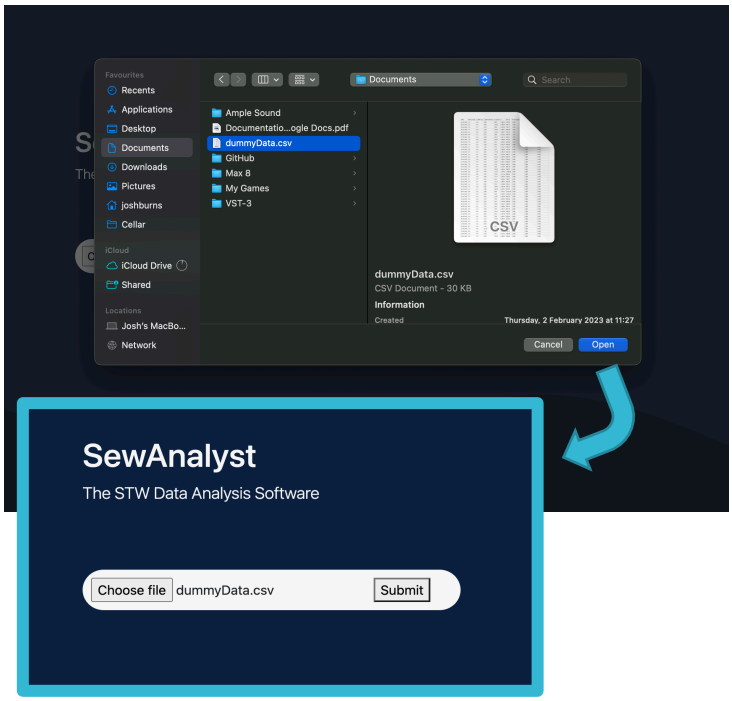
2) System Feedback Box



The system feedback box provides relevant info on the analysis of data submitted. Any errors that occur throughout the process of software usage will be displayed to the user here. Other messages will display if an error has occurred at either the data input phase, or the data processing phase. If either of these are detected, a more detailed analysis of the issue will be displayed within the console.

3) File Input Section

To input a file into the software press, the button labelled "choose a file". This will open your computer's default file browser. From here, navigate to your file and select it. If successful, the text to the right of the button will display the name of the file you have just selected.

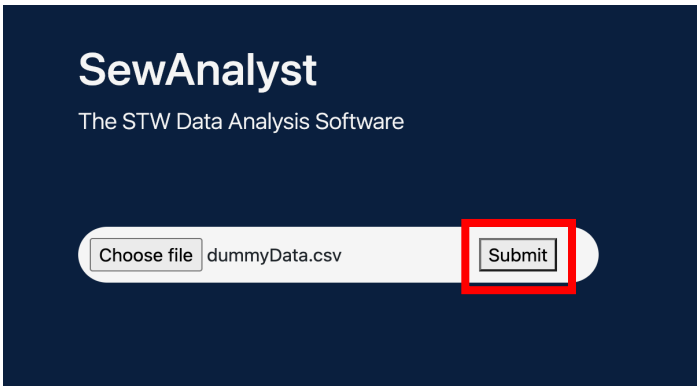


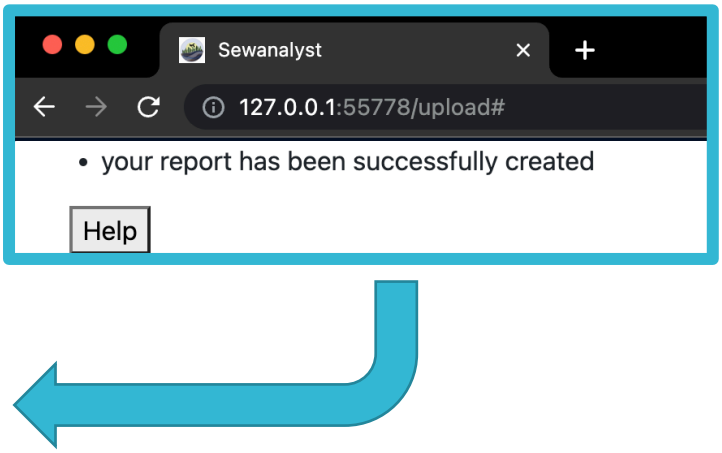
Please Note:

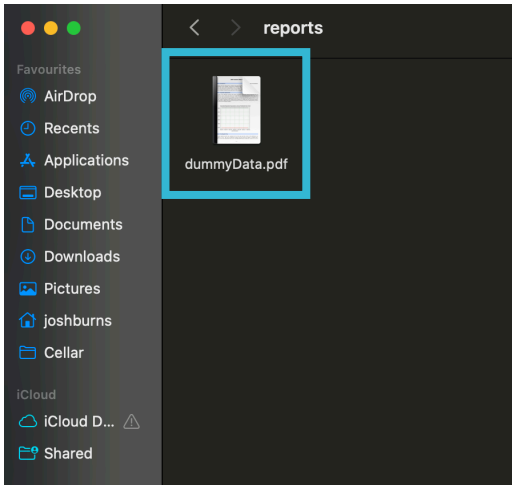
The format of your file is very important for the accuracy and validity of calculations performed by ***SewAnalyst***. To ensure it is correct please follow the **Data Formatting Guide** located in the next section of this document.

4) Submit Button

Finally, the submit button will be usable after a file has been selected. Once pressed, the process of analysis and report generation begins automatically. Any issues with the process will be displayed within the system feedback box. Once complete a message stating that the report has been generated will be displayed and the report will be saved to the reports folder within the SewAnalyst program folder.







3.3 - DATA FORMAT:

For SewAnalyst to perform accurate and valid analysis, data must be inputted in a specific format in the CSV file type. Below is the format in which data must be structured. This data can be tabled in any modern spreadsheet software that allows for export to different file-types.

Date	Population	G	I_DWF	E	DWF (reported)	FFT	PE (reported)	PE (forecasted)
01/01/21	-	-	-	-	-	-	-	-
02/01/21	-	-	-	-	-	-	-	-
03/01/21	-	-	-	-	-	-	-	-
…	…	…	…	…	…	…	…	…

- Date: Date data was collected
- Population: Population catchment of STW
- G: Per Capita Domestic Flow (l/hd/d)
- IDWF: Dry Weather Flow Infiltration (l/s)
- E: Trade Effluent Flow (l/d)
- DWF (reported): Dry Weather Flow as Reported by STW (l/d)
- FFT: Flow to Full Treatment (l/d)
- PE (predicted for the time frame)
- PE (actual value for the time frame)

Note: Data is not required for every day of the year for the software to perform analysis, however the accuracy of its findings will be dependent on the amount of data it is given.
Once data is in the format, it must be exported as a CSV file (Comma-Separated Value). To do this in Microsoft Excel do:

File → Save As → Save As Type → CSV

Once you have the CSV for the data you wish to upload, you may enter it directly into the software.

For more information, please visit this site: https://support.microsoft.com/en-us/office/save-a-workbook-to-text-format-txt-or-csv-3e9a9d6c-70da-4255-aa28-fcacf1f081e6

3.4 - REPORT FORMAT:

The automatically generated pdf report will be found within the directory, folder, as defined above. It is saved under the name of the csv-file used to generate the report. To view the pdf, simply double click on it, or alternatively use **right click → open**. Based on the results of the analysis this pdf may contain one or more pages. The general structure of the PDF document is defined below:

Example Report:

STW Analysis Report

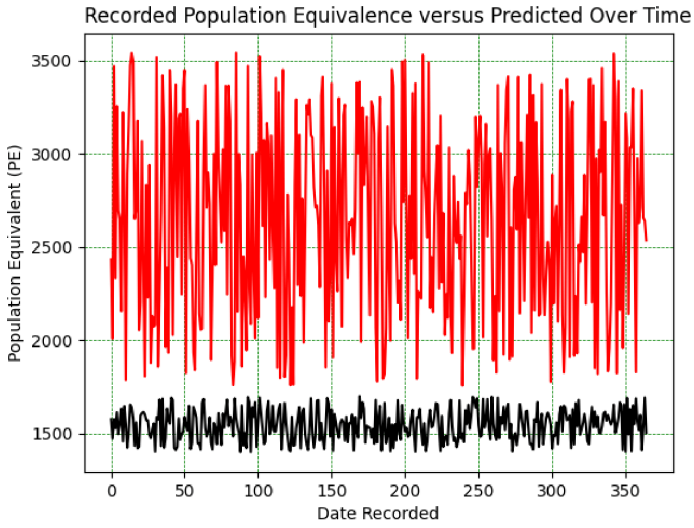
Section 0 : Introduction

The analysis of the data input has been conducted. The analysis will provide answers on: 1. Asses the accuracy and correctness of the company given PE for individual STWs 2. Verify that the company accurately calculated and reported their proper DWF for the according STWs 3. Verify that the company accurately calculated and reported the proper FFT for the according STWs 4. Qualitatively assure companies are ensuring provision of adequate sewage treatment capacity at water companies STWs

Introduction to analysis performed and meaning of analytical outcomes.

Section 1 : Population Equivalence

The severity of the difference between the reported population equivalence and the forecasted one was determined as "severe". This means there was a high difference between the reported PE and the actual forecasted one. This could insinuate the company is trying to minimise the maintenance cost for the STWs and as well may be hiding insufficient appropriate infrastructures for their respective true populations. This is a red flag, and it would be suggested to look further into the company as to whether they are being honest and they are just struggling to update their figures quick enough and accordingly or if they are lacking honesty and are trying to cut cost in an unethical manner.



Assessment of population equivalence:

The main aim of this section is to highlight the discrepancy between forecasted and actual population equivalence values for individual STWs throughout different recordings within the year. The graph visually represents this.

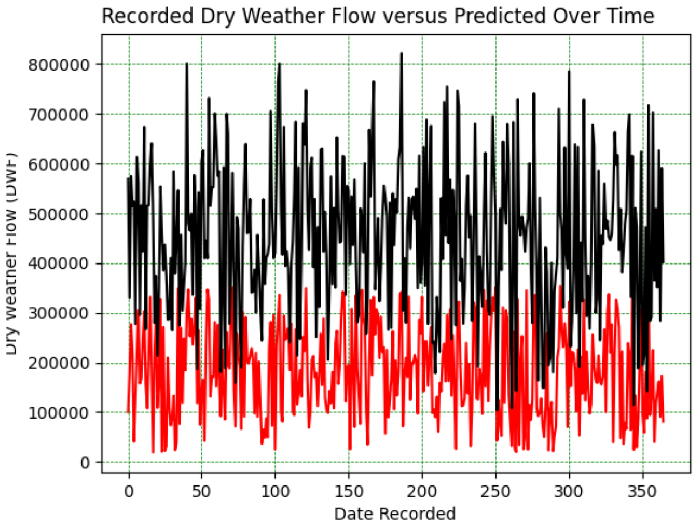
Section 2 : Dry Weather Flow

DWF value reported and the one calculated were not the same and the severity of this difference was determined as "mild". Due to a change in 2012, there is a lot more leeway about the calculations of this therefore could explain the mild difference reported. As well numbers may have been rounded up and

Page 1

STW Analysis Report

therefore vary the final value slightly or it was that a company was slightly over a threshold and wish to stay under, so they do not have to increase expenses on their STWs.



Assessment of Dry-Weather-Flow:

This section focuses on the differences of predicted ***DWF*** values based on their maximum potential values, reported values and predicted values. The focus ***STW*** is then graded based on this calculated deviation.

Section 3 : Flow to Full Treatment

FFT value reported and the one calculated were not the same and the severity of this difference was determined as "severe". Numbers and values used in its calculations may have been rounded up and therefore vary the final value slightly. As well there are two different ways to calculate this and therefore the other equation may have been used in which values vary and it is dependent on PE which if flagged earlier may be the issue. Alternatively, this raises an important warning about the company changing values in order to avoid certain requirements that comes with higher FFT which would mean an increase in cost for STW maintenance. There is a possibility the wrong data or something was forgotten during the calculations, but this is highly unlikely. As well, the common agreement is that FFT is around 3 times DWF however many companies disagree and say it is around 2 times only and this is where a lot of the differences come from.

Assessment of Flow to Full Treatment

Section 4 : Company Analysis

The overall difference severity was "mild". Some data values were flagged, or they were all averagely flagged for being mild differences meaning that some areas of data the company sent in did not match up with what was calculated or forecasted. This may just be miscalculations and a mistake by the company or it is an issue and they are trying to stay below the threshold to cut cost however this can be mildly detrimental to the environment and river ecosystems.

Overall Assessment of Company

STW Analysis Report

Section 5 : Conclusion

Overall to summarise the analysis of the data found:

PE: severe
DWF: mild
FFT:True
Company Analysis: mild

Conclusion highlighting the key grading for each section covered.

Troubleshooting:

If at any point the software fails to perform proper analysis or fails to give a conclusive result, this will be indicated either within the report content reading "There was an error with the [the analysis that failed] data analysis, please retry it and make sure your data is proper" or via the UI of the software and terminal output. In either case, the user should analyse the issue and ensure their data matches the required format and rerun the program to generate a new report.

Plot Interpretation:

This section outlines how to interpret the figures within the PDF report. It is important to remember that the quality of analysis depends solely on the quality of data inputted into the program. Additionally, figures on the report are accompanied by explanations of the findings. Below is a clear comparison between a 'good' and a 'bad' plot in accordance with **EA** guidance.

*In these figures, the **BLACK** line represents actual data and the **RED** line indicated predicted/forecasted data.*

Dry Weather Flow

The following are plots visualising the difference between EA guidance on Dry Weather Flow (DWF) and the values the STW have reported.

Exemplar DWF plot	Negative DWF plot

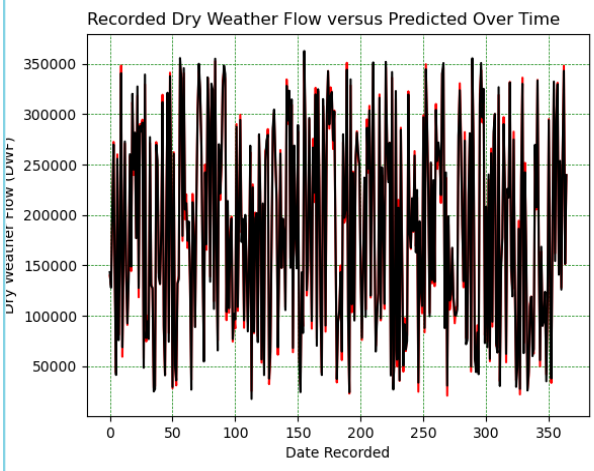
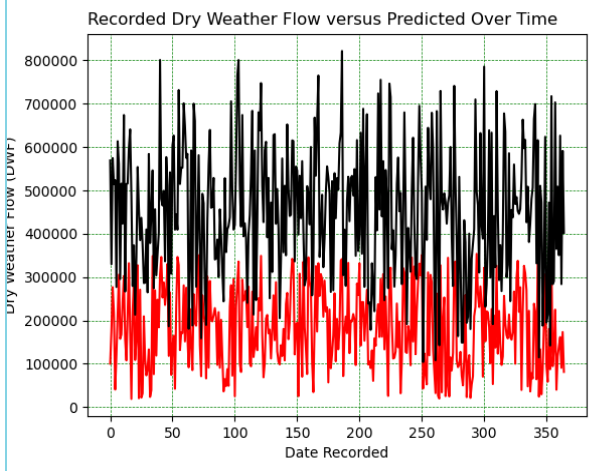

	

Figure 1.1	**Figure 1.2**
In this 'good' plot, **Figure 1.1**, the EA's forcasted prediction is extremely similar to the reported figures by the **STW**, as visualised by the two separate plots lying in the same plane and almost overlapping. This is an indication of good adherance to EA guidelines and suggests that the STW's reporting is accurate and legal.	However, in this 'bad' plot (**Figure 1.2)** the two lines are clearly distinct and differ severly, with the EA's predicted data lying almost completely below the actual data. This shows that the **STW** has not followed perscribed **EA** guidance and could indicate possible illegal activity.

Population Equivalent

The following are plots visualising the difference between EA guidance on Population Equivalent (PE) and reported values from the STW.

Exemplar PE plot	Negative PE plot

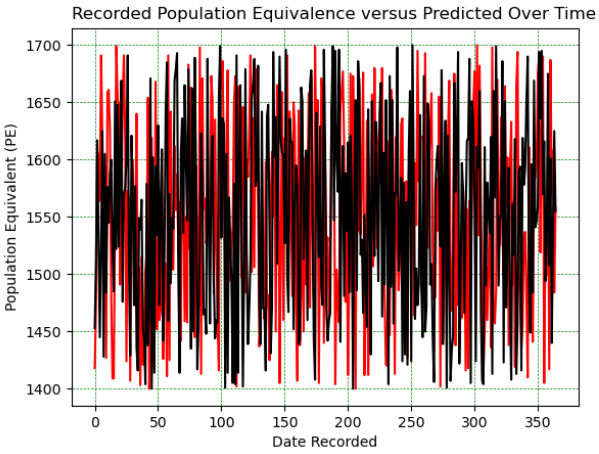
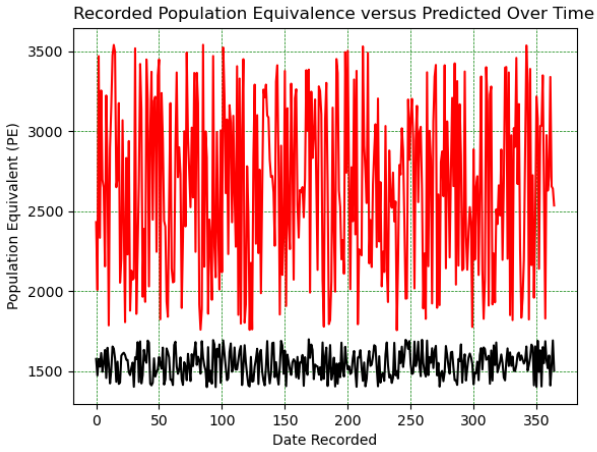

	

Figure 2.1	**Figure 2.2**
Visually, the predicted PE values and the line showing reported values on **Figure 2.1** are very similar and occasionally they overlap completely. This is indicative of tight adherance to guidelines in terms of Population Equivalent.	In **Figure 2.2** however, the reported PE data from the STW and the predicted values calculated from EA guidance are completely distinct, with no overlap and severe difference. This suggests that illegal activity from the STW.

Please note:

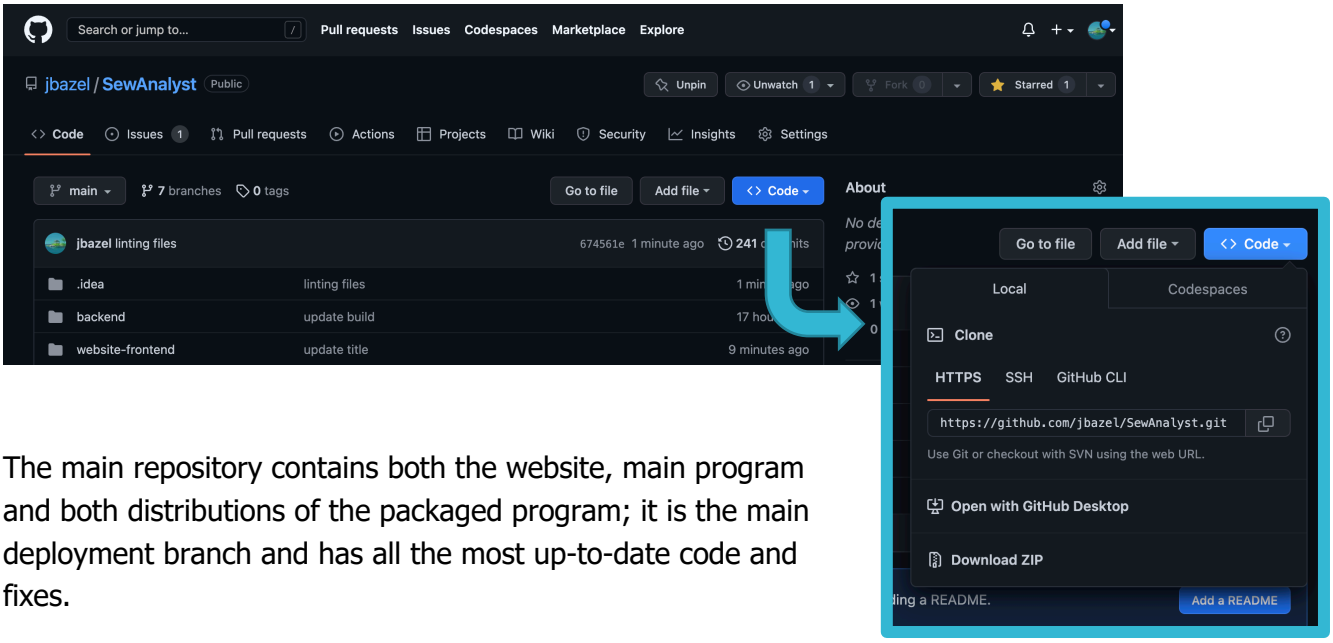
As the plots are entirely dependent on the quality of data inputted, the generated graphs may not look the same as these examples. Please use these exemplar plots as a guide in tandem with the written assessments outlined on the PDF to inform judgements on a potential environmental impact and the overall assessment of the company.

4 – SETUP/HOSTING

4.1 - ACCESSING THE CODE:

To access the SewAnalyst code please download from our GitHub page at:
https://github.com/jbazel/SewAnalyst

Once on the GitHub page you can navigate to the code button and either download as .zip file or open with GitHub desktop:



The main repository contains both the website, main program and both distributions of the packaged program; it is the main deployment branch and has all the most up-to-date code and fixes.

4.2 - HOSTING THE WEBSITE:

In order to host the website, we recommend using PaaS (Platform as a Service) due to the Express API functionality within the website. Once a hosting service is selected (such as Microsoft Azure [1] or AWS[2]), the website framework and code may be added to the remote server files. The node environment must then be set to "production" in order to ensure security. This can be set in the package.json file, added this line to the npm start function:

```
"name": "website-frontend",
"version": "1.0.0",
"description": "",
"main": "server/script.js",
"scripts": {
    "start": "NODE_ENV=production node server/script.js",
    "test": "jest"
```

The server may then be started as normal using "npm start" in the server terminal.

5 - WEBSITE IMPLEMENTATION DETAILS

5.1 – OVERVIEW

The website is built using Express, a minimal and flexible Node.js web application framework. The server is written in nodejs to provide JSON through REST API. The client side uses static HTML pages to load dynamic JSON content from the server via AJAX.

5.2 - IMPORTANT WEBSITE FUNCTIONS

index.js

Line number	Function name / purpose	Parameters	Description
2-10	downloadSoftwareWin	n/a	When the 'Download software for Windows' button is clicked, this function retrieves '/program_download-win' from the server.
31-40	routes	n/a	Defines an array of route objects, containing properties: 'path' and 'getView'. The 'path' property defines the URL path for the route. The 'getView' property is a function defined below.
34, 38	getView	params (extracted from the URL path)	getView takes a 'params' object as an argument, and returns a view for the corresponding path which is then displayed to the user.
43-77	changeLocation	n/a	This function is triggered when the URL changes in the web application. Depending on the size of the routePathSegments array, the function either loads the reports page content, or loads the list of reasons. It is done like this, so that it could be expanded for multiple other routes.
86-100	viewReportsPage	n/a	Fetches '/reportList' from the server. The response will be of the type JSON. This function then calls the 'allReports' function to get the html for the page.
102-126	reportReasonsPopup	reportName	It retrieves a JSON response from the '/reportReasons/:reportName' endpoint. If the response is empty, it displays a message using the 'noReasons' function. Otherwise, it displays a table of report reasons using the 'reportReasonsTable' function. Then, it returns the generated HTML.
131-147	allReports	data	Takes an object of report data and generates a HTML table. It calls the 'report' on each array in the object in order to

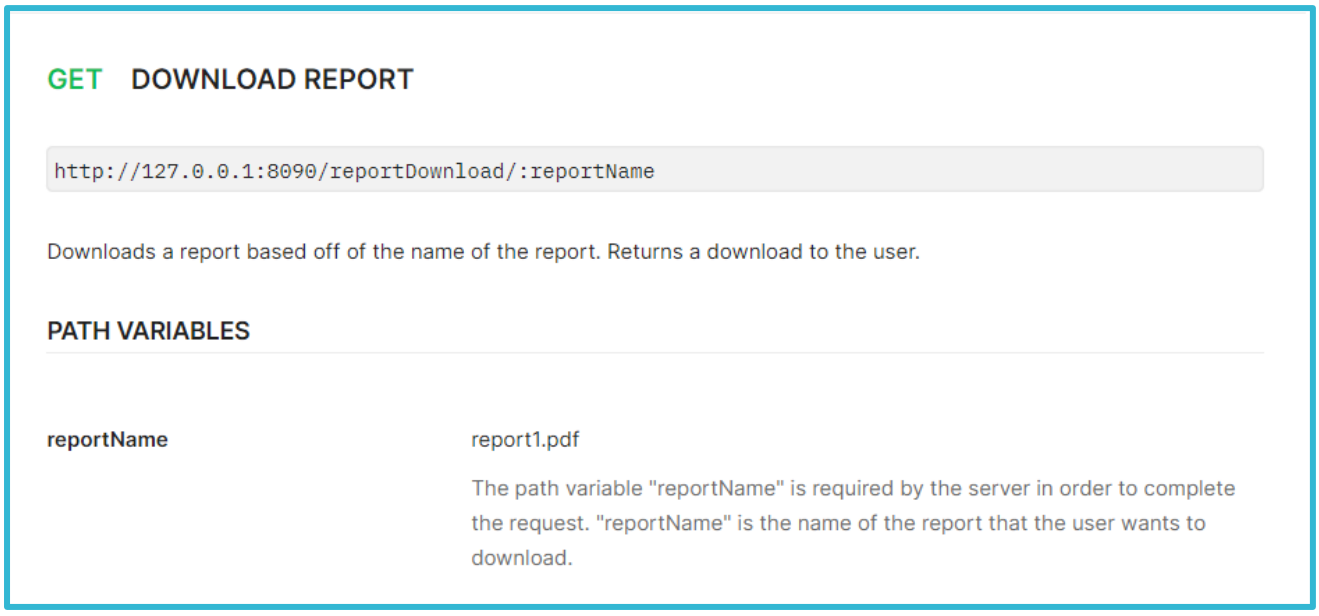
			populate each row of the table. It then returns the HTML for the table.
150-163	report	data	Generates each row to populate the table. Containing: a report's name, number of times reported, and buttons to flag and download the report.
207-224	generateForm	data	Generates a form with fields to report a reason for flagging a report. The form is displayed on the page when called. When the submit button is clicked, the form data is sent to the server for processing.

app.js

Line number	Function name / purpose	Parameters /	Description
24-32	saveData	file	Saves JOSN data to a file using the 'fs' module in Node.js. It takes a file as input, converts to JSON format and writes it to the "flaggedReports.json" file.
163-174	updateJsonfile	reportName	The function creates a data object with information about the report, including its name, the date it was made, an empty array for reviews, and a report counter. It then adds the data object to a list of all reports and saves the updated list to the JSON file.

5.3 - API DOCUMENTATION

Documentation completed using Postman (https://www.postman.com/)



Example Request DOWNLOAD REPORT

curl

```
curl --location 'http://127.0.0.1:8090/reportDownload/report1.pdf'
```

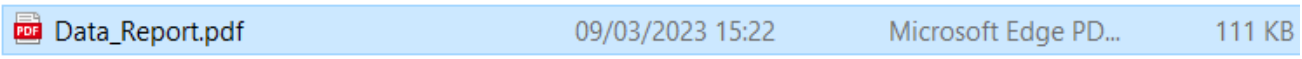
Example Request DOWNLOAD REPORT

curl

```
curl --location 'http://127.0.0.1:8090/reportDownload/report1.pdf'
```

This function downloads a file with the given **reportName** from the server's **files/reports** folder when requested and sets the last-modified header for the file.

Response found in download folder:



GET **DOWNLOAD PROGRAM**

`http://127.0.0.1:8090/program_download`

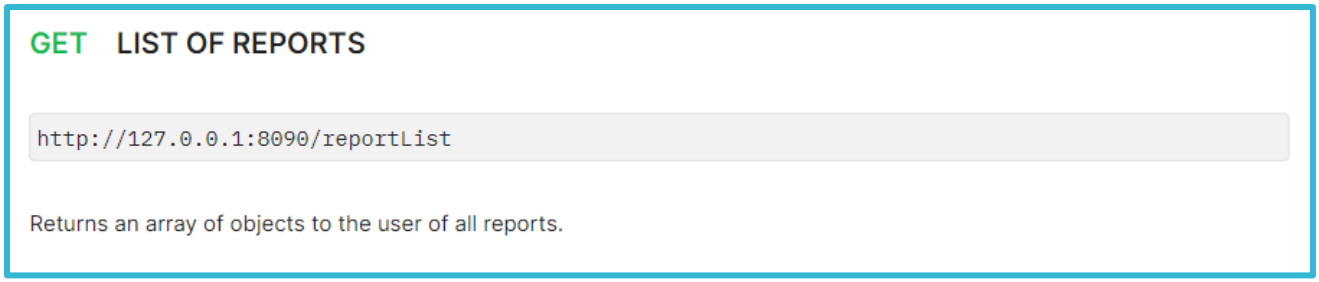
Downloads the software. Returns an executable file to the user.

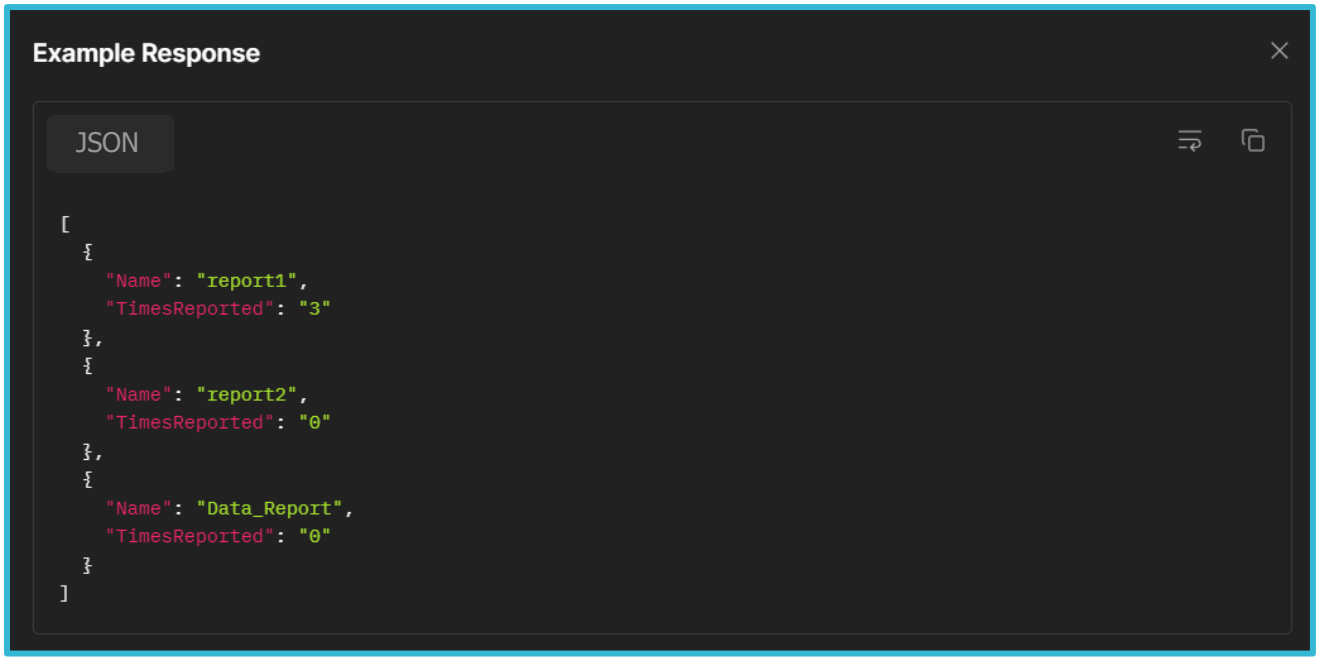
Example Request PROGRAM DOWNLOAD

curl

```
curl --location 'http://127.0.0.1:8090/program_download'
```

This function downloads the program to the user's device and the console logs that the download is taking place. The code also sets the Last-Modified header of the download response to the current time to stop http 304 responses.





Iterates through a list of all reports of extract each report's name and number of times it has been flagged. It then adds the name and report count as an object to the reportList. Finally, it sends the reportList as a response to the user.

GET REPORT REASONS

```
http://127.0.0.1:8090/reportReasons/:reportName
```

Returns an object of reasons for a particular report, based on the reportName sent by the client.

Plain Text

```
{
    "reportDate": ${reportDate},
    "reportReason": ${reportReason}
}
```

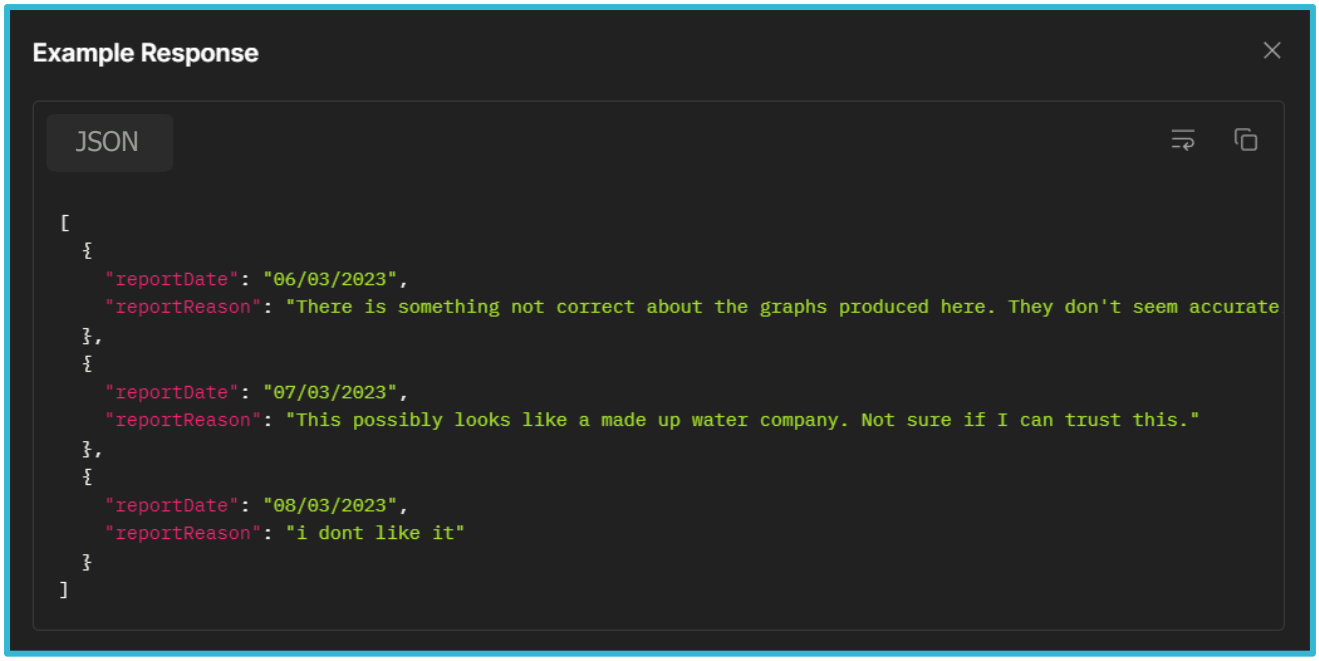
PATH VARIABLES

reportName	report1 The path variable "reportName" is required by the server in order to complete the request. "reportName" is the name of the report that the user wants to view the flagging log for.

Example Request New Request

curl

```
curl --location 'http://127.0.0.1:8090/reportReasons/report1'
```



When a request is made, the code extracts the report name from the request parameters and searches through a list of all reports to find the corresponding report. If the report is found, the code extracts a list of all reasons the report has been flagged and sends it as a response to the user.

POST FLAG REPORT UPLOAD

```
http://127.0.0.1:8090/flagReport/:reportName
```

Adds an entry to the reasons section of the JSON file for the particular report specified by reportName. Redirects the user to the home page and therefore returns the HTML of the homepage.

PATH VARIABLES

reportName	The path variable "reportName" is required by the server in order to complete the request. "reportName" is the name of the report that the user wants to send a flagging form for.

Body raw (json)

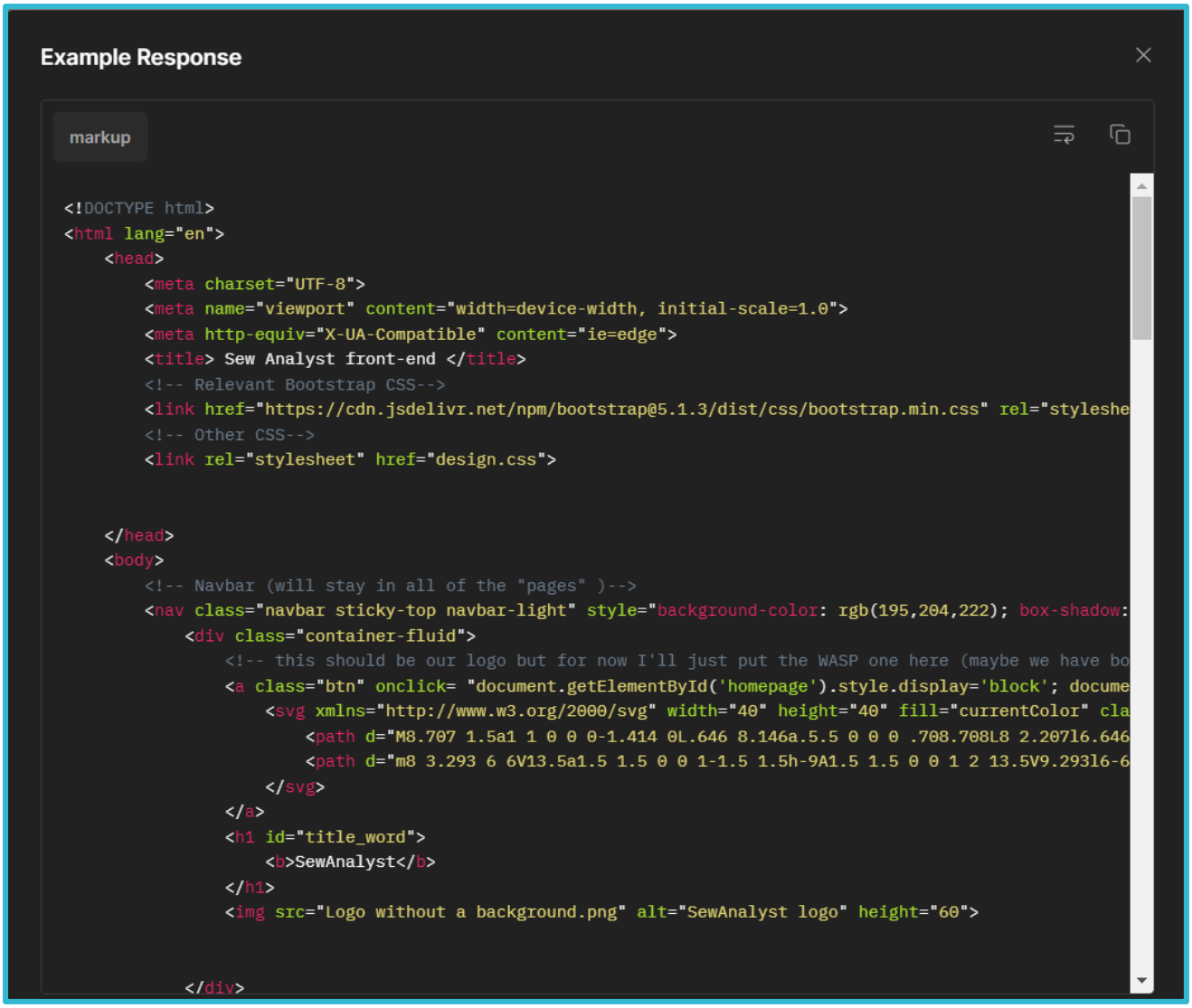
json

```
{
    "reason": "This data looks suspicious"
}
```

Example Request FLAG REPORT UPLOAD

curl

```
curl --location 'http://127.0.0.1:8090/flagReport/report2' \
--data '{
    "reason": "This data looks suspicious"
}'
```



When a request is made, the code extracts the report name and reason for the report from the request body and creates a new report object with a report date and reason. It then searches for the report in a list of existing reports, adds the new report object to the list, and increments the report counter. Finally, it saves the updated data and redirects the user to the application's home page.

POST **UPLOAD REPORT**

`http://127.0.0.1:8090/upload`

Once the user has selected a report from their device, this post request will upload it to the website.

Example Request UPLOAD REPORT

curl

```
curl --location 'http://127.0.0.1:8090/upload' \
--form 'sampleFile=@"/C:/Users/jasmi/Documents/Software Engineering/STW1.pdf"'
```

Example Response

markup

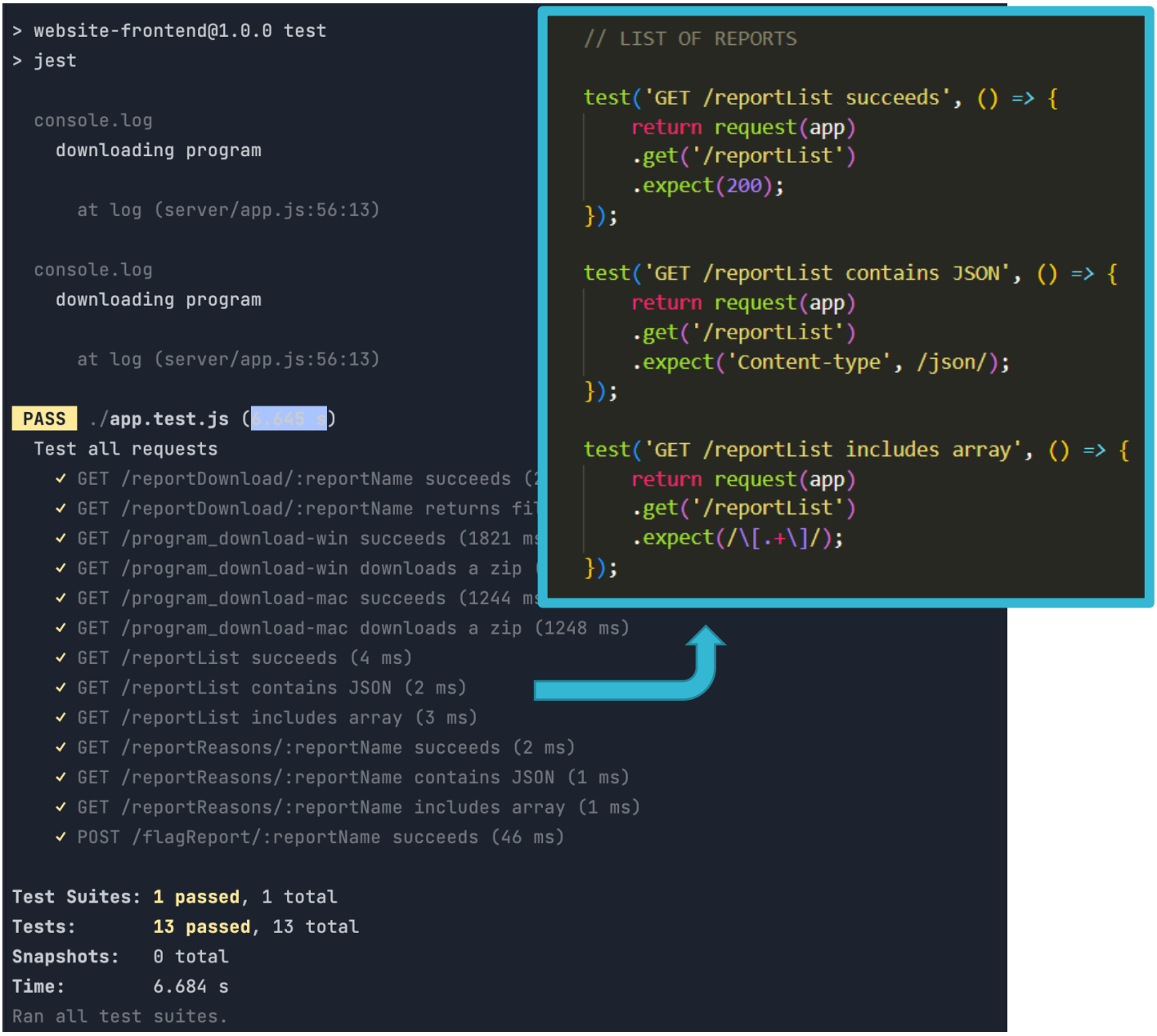
```
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <meta http-equiv="X-UA-Compatible" content="ie=edge">
        <title> Sew Analyst front-end </title>
        <!-- Relevant Bootstrap CSS-->
        <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css" rel="styleshe
        <!-- Other CSS-->
        <link rel="stylesheet" href="design.css">


    </head>
    <body>
        <!-- Navbar (will stay in all of the "pages" )-->
        <nav class="navbar sticky-top navbar-light" style="background-color: rgb(195,204,222); box-shadow:
            <div class="container-fluid">
                <a class="btn" onclick= "document.getElementById('homepage').style.display='flex'; documen
                    <svg xmlns="http://www.w3.org/2000/svg" width="40" height="40" fill="currentColor" cla
                        <path d="M8.707 1.5a1 1 0 0 0-1.414 0L.646 8.146a.5.5 0 0 0 .708.708L8 2.207l6.646
                        <path d="m8 3.293 6 6V13.5a1.5 1.5 0 0 1-1.5 1.5h-9A1.5 1.5 0 0 1 2 13.5V9.293l6-6
                    </svg>
                </a>
                <h1 id="title_word">
                    <b>SewAnalyst</b>
                </h1>
                <img src="Logo without a background.png" alt="SewAnalyst logo" height="60">
```

The function extracts the uploaded file and sets the path to save it to. It then moves the file to the specified path and calls the '**updateJsonfile**' function. Finally, it sends a success status and redirects the user to the application's home page.

5.4 - API TESTING

Jest testing provides a framework for automated testing of HTTP requests and responses. It has been used to write test cases for the API endpoints. In each test we monitor both HTTP responses as well the response times.

The test cases can be run using the command "npm test" in the command line. These are the results of the testcases along with code for a section of the test cases:



5.5 - DATA STRUCTURE

The report data is stored in a JSON file, with the report files themselves being uploaded to a separate folder in the system. This file is stored in the "data" folder within the website file directory.

Attributes of JSON file:

Attribute Name	Description	
reportName	The report name is the name of the pdf document taken upon report upload. It is how the report is referred to throughout the other sections of the website.	
dateOfReport	The date when the report was uploaded to the website.	
reviews	The reviews contain the data submitted by the user when they flag a report. Stored as an array of objects. Each object contains two more variables: reportDate, reportReason.	
	reportDate	The date that the "review" was submitted.
	reportReason	The reason the user gave as to why they were flagging the report.
reportCounter	This stores the total amount of times a report has been flagged.	

When the post request is made to upload a report, the function "updateJsonfile" is called, updating all JSON report attributes.

```
function updateJsonfile(reportName){
    let reportDate = new Date().toLocaleString()
    reportDate = reportDate.split(",")[0]
    const data = {
        reportName: reportName,
        dateOfReport: reportDate,
        reviews: [],
        reportCounter: "0"
    };
    loadReports.push(data);
    saveData(reportFolderPath);
}
```

reportName is the variable passed into the function, which corresponds to the .name property of the file.

The **dateOfReport** is calculated using the standard Date() function.

The **reviews** variable is set to an empty array initially, so it can be populated with objects at runtime.

The **reportCounter** is initialised to be "0" in preparation to be updated during runtime.

6 - APPLICATION IMPLEMENTATION DETAILS

The main application consists of a flask frontend UI constructed from the HTML/css/JS webstack within the file "main.py". All analysis and report generation comes from the functionality defined in the "library.py".

6.1 - APPLICATION FRONTEND

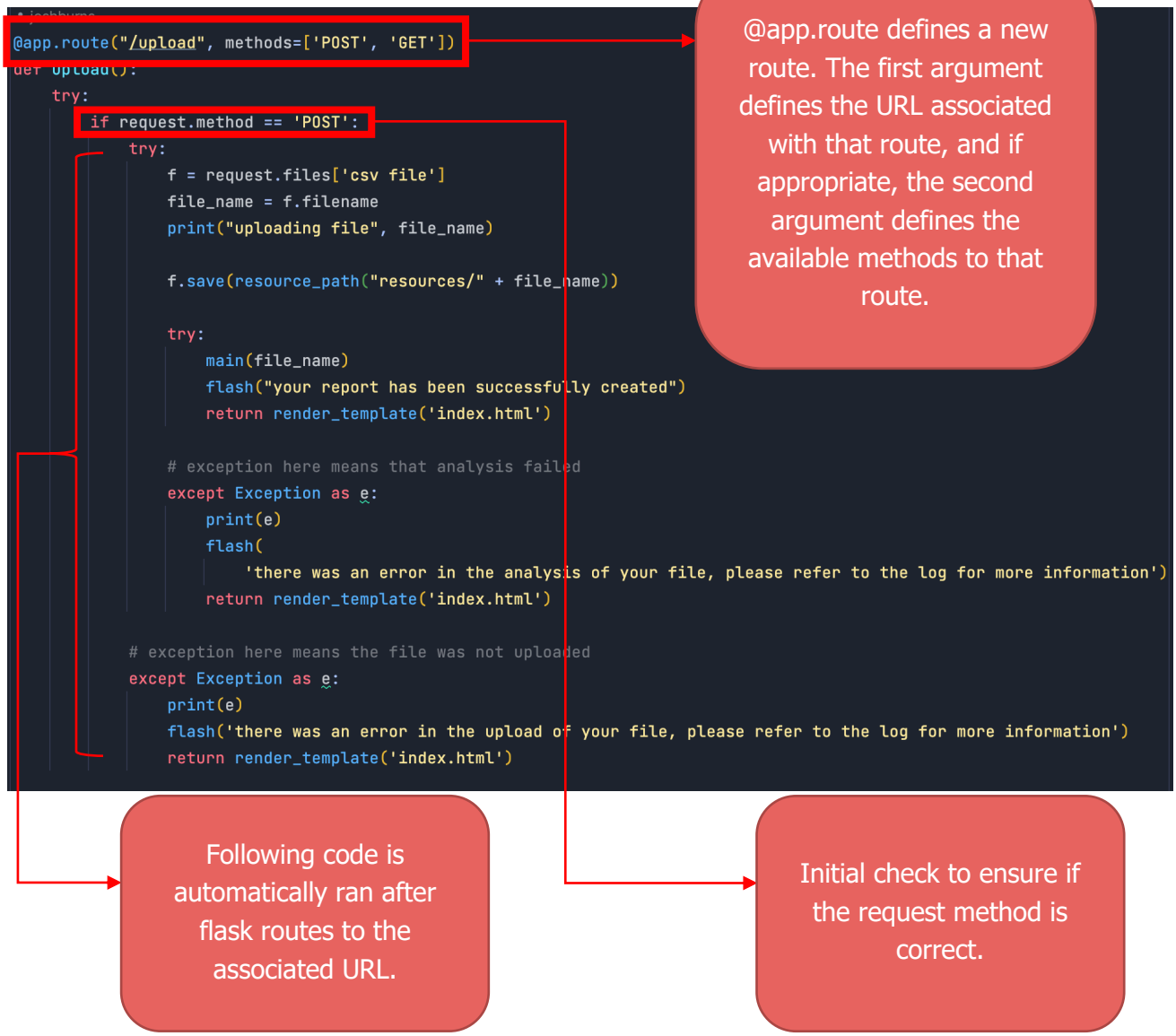
The frontend of the SewAnalyst software, implemented in "main.py", is built on the Flask Framework. The UI is composed of the HTML/css web stack and is rendered through Flask within the web browser. Routing is used to assign URLs to functionality within the program with the "home" being ran upon first launching the program and rendering the "index.html" template.

```
joshburns +1
@app.route("/")
def home():
    try:
        print("routing to home, attempting to render template")
        return render_template('index.html')
    except Exception:
        print("render template error")
```

The main section of the frontend comes in the "/upload" route. This is route actioned by the submission of the HTML form within the UI and immediately calls the associated functions from "library.py" to perform data analysis and report generation.

```
<div class = "container2">
    <form action = "/upload" method = "post" enctype = "multipart/form-data">
        <input type="file" name="csv file"  placeholder="csv file" value="{{ request.form['csv data'] }}">
        <button type="submit">Submit</button>
    </form>
</div>
```

Validation that file uploaded is of type multipart/form-data.



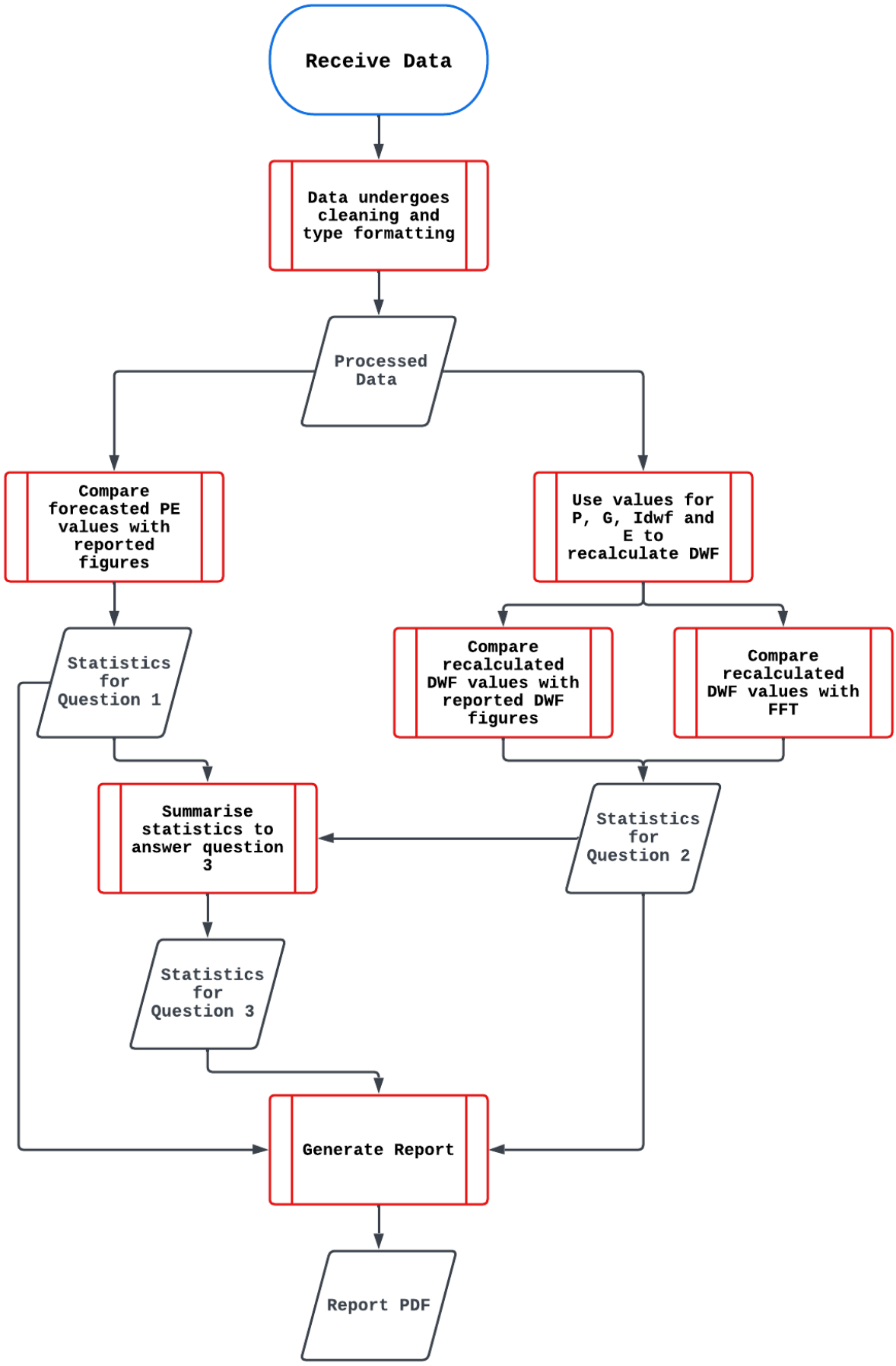
The try/except methods shown above are structured hierarchically in order to catch all errors, as well as where they occur within the process of data upload to report generation. This allows the "flash" functions to deliver relevant messages to the HTML UI for the user to view.

```
{% with messages = get_flashed_messages() %}
    {% if messages %}
        <ul class=flashes>
        {% for message in messages %}
            <li>{{ message }}<a href="#"  data-dismiss="alert" aria-label="close">×</a> </li>
        {% endfor %}
        </ul>
    {% endif %}
{% endwith %}
{% block body %}{% endblock %}
```

6.2 - APPLICATION BACKEND

After an appropriate csv file has been uploaded via flask, it undergoes the following processing and analysis by functions in library.py:

Function Name	Parameters	Parameter Descriptions	Description
processData	filePath	The path to the file uploaded via the Flask frontend. This is calculated relatively based on the application's location.	The csv file is converted into a panda dataframe. Its contents are then checked and validated to ensure all necessary attributes are present. Missing values are accommodated for, and values are converted into the correct format.
calculateDWF	data	The processed and validated dataframe generated by the *processData* function.	Using the relevant fields, the value of DWF is recalculated using the DWF formula from section [x].
calculateDifference	Forecast, reported	Calculates on the pairs: • (forecasted PE, reported PE) • (forecasted DWF, reported DWF)	Determines the presence and severity of significant differences between forecasted and reported figures.
fftThreshold	forcastedDWF, FFT	The forecasted values for DWF as calculated by the *calculateDWF* function as well as the data from *processData* for FFT.	The reported FFT is compared with the recalculated DWF values to determine if FFT coverage is adequate.
questionThree	statistics	The statistics generated by *calculateDifference* and *fftThreshold*	Statistics are coalesced to produce an answer for Q3.
generateFigures	statistics	The statistics generated by *calculateDifference* and *fftThreshold.*	The data for PE and DWF are plotted onto line graphs for comparison.
generateReport	Statistics, figrures	The statistics generated by *calculateDifference, fftThreshold* and *questionThree* as well as the path to the figures generated by *generateFigures.*	Uses the generated statistics and graphs to generate a pdf report outlining the findings of the analysis and their consequences.


Receive Data
Data undergoes cleaning and type formatting
Processed Data
Compare forecasted PE values with reported figures
Use values for P, G, Idwf and E to recalculate DWF
Statistics for Question 1
Compare recalculated DWF values with reported DWF figures
Compare recalculated DWF values with FFT
Summarise statistics to answer question 3
Statistics for Question 2
Statistics for Question 3
Generate Report
Report PDF

The functions above are coalesced into a main function, which, during runtime is called through the flask routing from within the file "main.py":

```
def main(file_name):

    FILENAME = file_name[0:-4]
    print(FILENAME)
    # reads the csv file in a panda dataframe, checks all required attributes are present
    # and cleans/formats the data
    data = processData(resource_path("resources/"+file_name))

    # assign required data fields to variables to be used for analysis
    dates = data['DATE']
    PEActual = data["PE_REPORTED"]
    PEForecast = data["PE_FORECAST"]
    DWFActual = data["DWF_REPORTED"]
    FFTActual = data['FFT']

    # calculate DWF using provided data and assign to the dataframe
    data['DWF_RECALCULATED'] = calculateDWF(data)
    DWFForecast = data['DWF_RECALCULATED']
    stats = {}

    # determines the significance and severity of difference between the SOLAR PE forecast
    # and reported PE figures
    stats['sigDifPE'], stats['difSevPE'] = calculateDifference(PEForecast, PEActual)
    # determines the significance and severity of difference between the recalculated DWF
    # figures and the reported DWF figures
    stats['sigDifDWF'], stats['difSevDWF'] = calculateDifference(DWFForecast, DWFActual)
    stats['FFTDiscrepancy'] = fftThreshold(DWFForecast, FFTActual)
    # Uses the statistics generated from the calculateDifference functions to determine an
    # answer for question 3
    stats['q3Discrepancy'], stats['q3Severity'] = questionThree(stats)

    # used by the developer to check if all functions have produced an appropriate output
    for statistic in stats.keys():
        print(stats[statistic])

    try:
        # use the provided data to plot PE and DWF graphs
        generateFigures(dates, PEActual, PEForecast, DWFActual, DWFForecast)
        print("FIGURES GENERATED")
        # use the generated statistics and graphs to create a pdf report
        generateReport(stats, resource_path("resources/PE_plot.png"), resource_path(
            "resources/DWF_plot.png"), FILENAME)
    except Exception as e:
        # output any error if one occurs in the figure generation or report creation phase
        print (e)
```

6.3 - DATA PROCESSING

Handled by: **processData(**pathToData**)**

This function reads the uploaded CSV file into a panda dataframe and assigns to it appropriate headers. The values in the date column are converted into datetime objects for the purposes of figure generation. Data validation and basic cleaning then occurs where numerical values are converted into floating point numbers for analysis. If an expected attribute is missing an alert is generated and the process is halted. In the case that a value is missing, the row in which this occurs is dropped and an alert is produced. The dataframe is then stored in a variable called 'data' which is returned for other functions to use.

6.4 - DATA ANALYSIS AND STATISTIC CREATION

6.4.1 - **CALCULATING** DRY WEATHER FLOW (DWF)

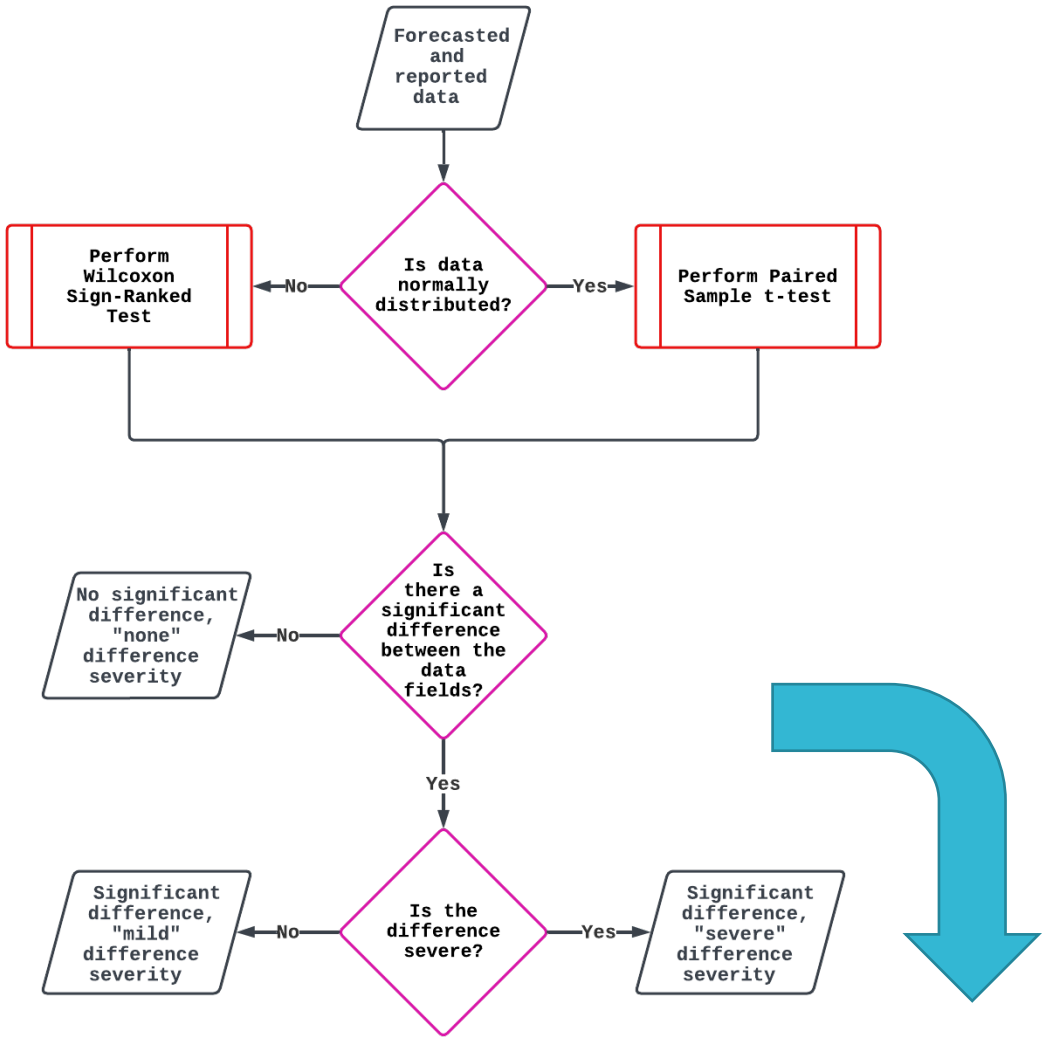
Handled by: **calculateDWF(**data**)**

From the data, the values for catchment population (P), per capita domestic flow (G), dry weather infiltration (Idwf) and trade effluent flow (E) are extracted and used to calculate DWF for each date provided using the following formula:
DWF = PG + Idwf + E
These values are then returned as a panda series object.

6.4.2 - **COMPARING** DATA FIELDS

Handled by: **calculateDifference(**forecastedData, reportedData**)**

To answer question 1, this function receives the SOLAR PE forecast and the reported PE figures. For question 2, the recalculated DWF values and reported DWF values are received.
First, it is determined if the values are normally distributed using the Shapiro-Wilk test. If this returns a p value less than 0.05, there is evidence to suggest that the data is not normally distributed. If the data can be assumed to be normally distributed ($p > 0.05$), a paired sample t-test is performed to determine if there exists a significant difference between the two sets of data. Otherwise, the Wilcoxon sign-rank test is performed. If the p value from either of these tests are less than 0.05, there is evidence to suggest that a significant difference is present between the datasets. To determine the severity of the difference (given that one is present), the value of cohen's d with respect to the two datasets is computed. If cohen's d is less than 0.8, the difference is deemed to be mild, otherwise it is considered severe. The function then returns whether a significant difference is present and the severity of the difference.




```python
# function for determining if differences between data are significant and how severe they are
def calculateDifference(forecastData, reportedData):
    # if there a significant difference between datasets and how severe is that difference
    significantDifference = False
    differenceSeverity = "none"
    # determine if data is normally distributed
    forecastNormal = stats.shapiro(forecastData).pvalue
    reportedNormal = stats.shapiro(reportedData).pvalue

    if (forecastNormal >= 0.05) and (reportedNormal >= 0.05):
        # if data is normally distributed, perform paired sample t-test
        sigDifTest = stats.ttest_rel(forecastData, reportedData).pvalue
    else:
        # if data is not normally distributed, perform wilcoxon sign-ranked test
        sigDifTest = stats.wilcoxon(forecastData, reportedData).pvalue

    if sigDifTest < 0.05:
        significantDifference = True
        # if there is a significant difference, calculate cohens d
        cohensD = (mean(forecastData) - mean(reportedData)) / (sqrt((stdev(forecastData) ** 2 + stdev(reportedData) ** 2) / 2))
        print(cohensD)
        # use value of cohens d determine the severity of the difference
        if cohensD < 0.8:
            differenceSeverity = "mild"
        else:
            differenceSeverity = "severe"
    return significantDifference, differenceSeverity
```

6.4.3 - **DETERMINING** FFT ADEQUACY

Handled by **fftThreshold(**DWFForecast, FFTActual**)**

This function receives the recalculated DWF figures from calculateDWF and the reported figures for FFT from the original dataset. First, the mean value for the DWF and FFT figures are calculated. Then the following comparison is performed:
If mean DWF multiplied by three is less than mean FFT, the treatment work has adequate FFT coverage, and the function returns False. Otherwise, FFT coverage is inadequate, and the function returns True.

6.4.4 - **ANSWERING** QUESTION 3

Handled by: **questionThree(**stats**)**

If there any of the comparisons of the PE, DWF or FFT figures return no significant difference, this function will return False. If all determine that there is a significant difference and are severe, True and "severe" are returned. However, if not all significant differences are severe, the function returns True and "mild". These encodings are used to produce an appropriate response in the generation of the report.

6.5 - FIGURE CREATION

Handled by **generateFigures(**dates, PEActual, PEForecast, DWFActual, DWFForecast**)**, present in figureGeneration.py

This function receives the data used for statistic creation to plot the differences between the forecasted and reported data across time on a pair of line graphs. The generated figures are saved as .png images for use in the report generation process.

Example function for generating PE plots:

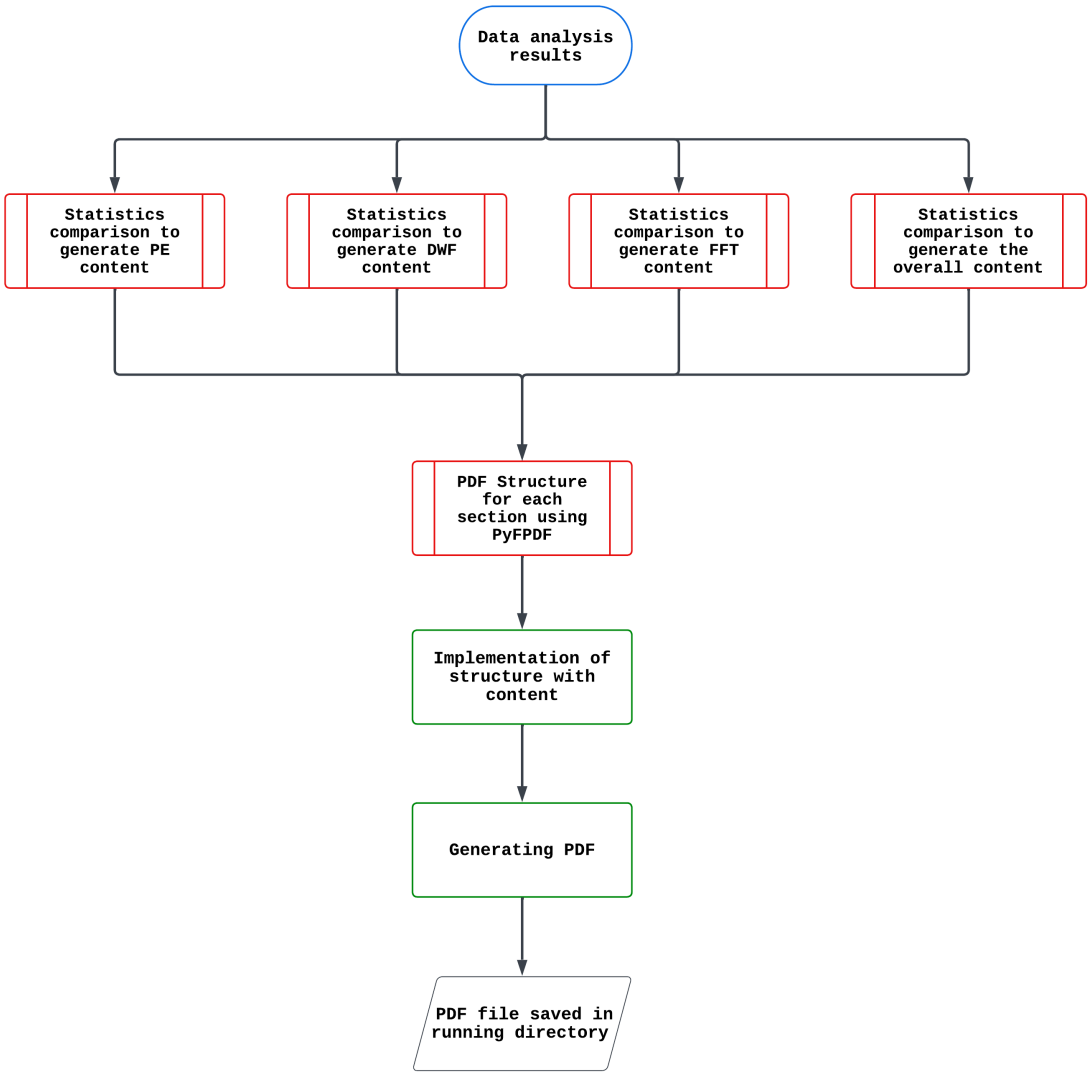
```python
# function used to generate line graph comparing SOLAR PE forecast with reported
# PE figures
def genFigPE(dates, PEActual, PEForecast):
    ## plot onto one graph
    plt.plot(dates, PEForecast, color='red', label='Predicted')
    plt.plot(dates, PEActual, color='black', label='Recorded')

    ##graph aesthetics
    plt.title("Recorded Population Equivalence versus Predicted Over Time", loc = 'left')
    plt.xlabel("Date Recorded")
    plt.ylabel("Population Equivalent (PE)")
    plt.grid(color = 'green', linestyle = '--', linewidth = 0.5)

    ##save as an image in the working directory
    plt.savefig('PE_plot.png')

    ##close
    plt.close()
```

The two datasets for **Population Equivalence** are plotted against each other. Arguments pertaining to "color" may be changed to fit user preference.

Graph visual aesthetics are set. These can be changed by editing the labels e.g. the "color" argument may be changed as well as the line style and line width.

6.6 - REPORT GENERATION

There are many steps to the report generation and two different libraries are used in order to generate the pdf, PyFPDF and PIL. The prerequisite data needed for the code to work is the severity results for each data analysis of PE, DWF, FFT and the overall severity. As well as the name of the graphs generated earlier. The first step is the selection of the appropriate content based on the results of the data analysis performed. The second step is the creation of the pdf and its structure and then integrate the appropriate content. The third step is the integration of the pictures of the graphs representing the data analysis. On the next page is a flowchart of the overall process.



Content Creation

The content of the pdf varies based on result of the data analysis and each data analysis is ranked with a difference severity of "none", "mild", "severe". With these different results 3 different paragraphs can be selected for each section of the report representing each data analysis conducted. These paragraphs contain explanations for the result. If the data analysis did not give any of the three above categories, then an error message is written instead.

```
def PEParagraph(PeSeverity):
    if PeSeverity == "none":
        paragraph = 'The severity of the difference
    elif PeSeverity == "mild":
        paragraph = 'The severity of the difference
    elif PeSeverity == "severe":
        paragraph = 'The severity of the difference
    else:
        paragraph = 'There was an error with the PE
    return paragraph
```

PyFPDF

PyFPDF is [1] an external library used to generate the pdf report for our analysis.
The PDF class has attributes and methods to facilitate the creation of each section of the report using the same structure and design. The class contains method implementations, functions, which constructs the header and footer of each page and they construct the title and text as well as any images for the appropriate content selected earlier. Example of which can be found below (Fig...)The class is then utilised to construct each section in the pdf and then the pdf is outputted, created, in the directory of which this is running. The details of PyFPDF methods used can be found in the documentation online if the user wishes to change any of the settings for the pdf structure [3].

```python
#Creating the pdf
class PDF(FPDF):
    def __init__(self):
        super().__init__()
        self.WIDTH = 210
        self.HEIGHT = 297

    def header(self):
        #Title
        self.set_font('Arial', 'B', 15)
        w = self.get_string_width(title) + 6
        self.set_x((210 - w) / 2)
        self.cell(w, 9, title, 0, 1, 'C', 0)
        self.ln(10)

    def footer(self):
        # Page numbers in the footer
        self.set_y(-15)
        self.set_font('Arial', 'I', 8)
        self.set_text_color(128)
        self.cell(0, 10, 'Page ' + str(self.page_no()), 0, 0, 'C')


    def titleAndText(self, num, title, txt):
        #Text Format
        self.set_font('Arial', '', 12)
        self.set_fill_color(200, 220, 255)
        self.cell(0, 6, 'Section %d : %s' %(num, title), 0, 1, 'L',
        self.ln(4)
```

Initialise as PDF and set width and height.

Set the header of the document including title.

Set the footer of the document including page number.

Set the page title and font settings for text within the document.

PIL

The PIL library [2] is used and imported to put an image in the pdf from the directory and modify its size so that it fits appropriately on the page if the image dimensions are bigger than the page. A shown billow PIL is used to retrieve the image size which is then checked against the page size. If it is too big then the image width and height are set to the page width and height and this is then adapted using a PyFPDF method image() setting the appropriate width and height.

```
def ImageSet(self, img):
    #Set up the image so it fits on the page
    iMage = Image.open(img)
    width, height = iMage.size

    width, height = float(width * 0.264583), float(height * 0.264583)
    pdfSize = {'P': {'w': 210, 'h': 297}, 'L': {'w': 297, 'h': 210}}
    if width < height:
        orientation = 'P'
    else:
        orientation = 'L'


    if width < pdfSize[orientation]['w']:
        width = width
    else:
        width = pdfSize[orientation]['w']
    if height < pdfSize[orientation]['h']:
        height = height
    else:
        height = pdfSize[orientation]['h']

    self.image(img, w=width,h=height)
    self.ln()
```

Image opened and size retrieved

Default orientation set and max image size defined (the size of the page)

Checking the image size fits on the page and adapting it if needed.

Implementing the image into the document

6.7 - PACKAGED PROGRAM

The python backend of our software (the downloadable distribution) is packaged as an executable file; no installation of python is required and SewAnalyst can run as soon as it has been downloaded. SewAnalyst is not directly cross-platform, so two separate distributions exist for Windows and for MAC OS.

7 - MAINTENANCE

7.1 - BUILDING THE SOFTWARE:

If any changes are made to the software, it will need to be re-built as an executable. To perform this, we are using the PyInstaller Library. Within the backend section of our program, you will see a file titled "main.spec". This is the file that contains the instructions for how to build the program – it is run via typing in terminal "PyInstaller main.spec" within the directory of the file "main.py".

When building, please ensure your python interpreter only contains relevant libraries in order to keep the file size of the packaged executable small. Any external python files linked to the main file will be detected automatically and added to the main executable. All relevant external libraries can be found within the requirements.txt file in the backend folder. In order to configure a new interpreter with only the correct libraries navigate to the backend folder and input these commands into the terminal:

```
python -m venv venv

source venv/bin/activate

pip install -r requirements.txt
```

This will create a venv (python virtual environment) that will contain only the relevant libraries for packaging SewAnalyst.

If at any point the software needs to be built from scratch (for example, if a platform of the software distribution if added) you will need to run the command "PyInstaller --onefile main.py". This will construct a "**dist**" and "**build**" folder, as well as a new main.spec file. The **dist** folder is the folder that will contain the main executable.

Note: *you may only build to the current system you are working on. This means, for example, to build the software for a windows machine, it must be built on a windows machine. If any software updates occur, the software will need to be built on the respective operating system for each distribution of the software.*

Please branch on the "backend" branch for each new distribution.

As the SewAnalyst program works by dynamically adding and accessing files in different paths, you will need to specify these resources within the "datas" argument of the spec file. It takes a 2-tuple of strings:

1. The location of the file currently in the directory
2. The location of the file at runtime.

As the executable code contains a function to calculate file path at runtime you may specify the same directory twice:

```
joshburns
def resource_path(path):
    try:
        base_path = sys._MEIPASS
    except Exception:
        base_path = os.path.abspath(".")
    return os.path.join(base_path, path)
```

```
datas=[('templates', 'templates'), ('static', 'static'), ('resources', 'resources'), ('reports', 'reports')],
```

A common issue that occurs in the build process are Hidden Imports. You will detect this if you try and run the executable and an ImportError is given. PyInstaller works via adding hooks to file paths for libraries in order to package them with the executable, however sometimes it is unable to hook onto second-level imports. In order to fix this issue, simply add the module to the "hiddenimports" argument of the spec file. This commonly occurs within the sklearn library.

```
hiddenimports=["sklearn.metrics._pairwise_distances_reduction._base",
"sklearn.metrics._pairwise_distances_reduction._datasets_pair",
"sklearn.metrics._pairwise_distances_reduction._middle_term_computer"],
```

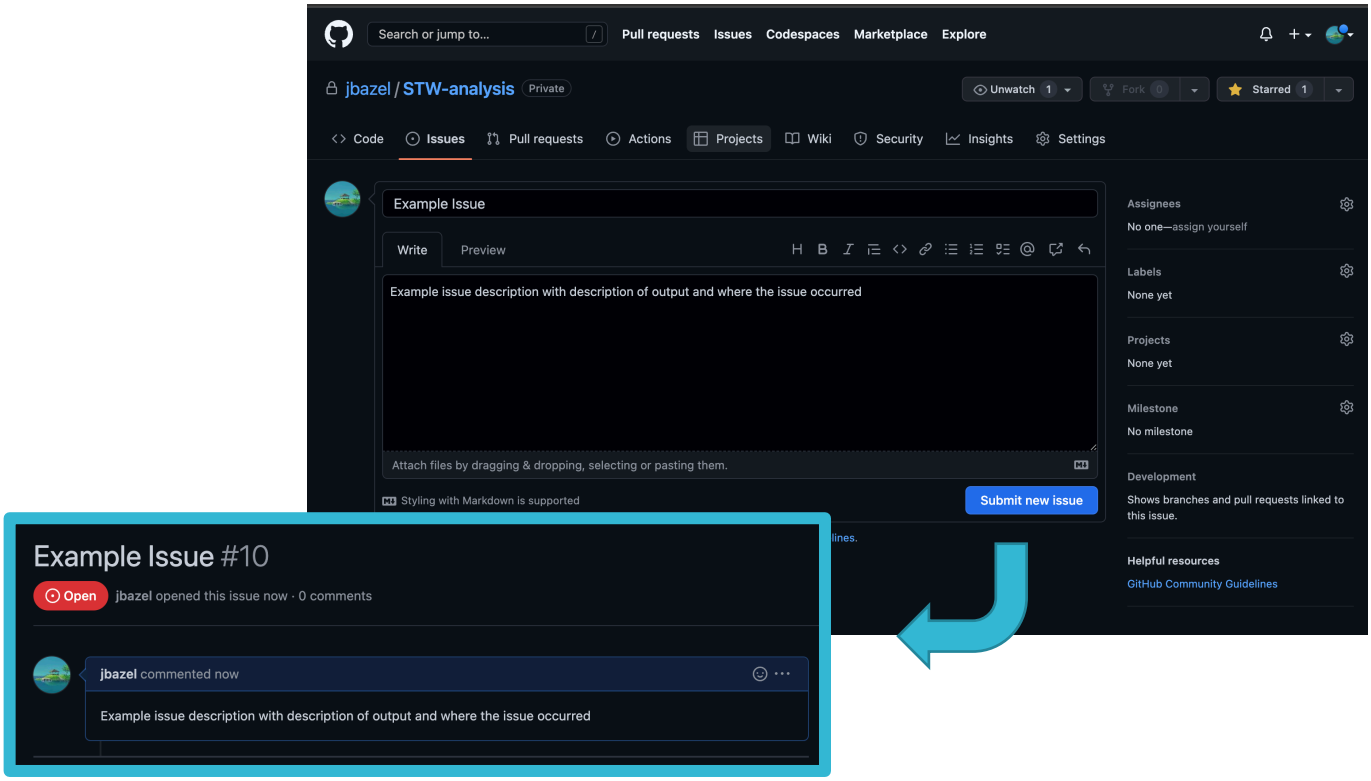
Once packaged, you will see the executable within the **dist** folder of the working directory. To now package the software ready for deployment, copy the following files into a new folder:

- **dist folder containing main executable**
- **reports**
- **resources**
- **static**
- **templates**

Once complete, this folder may be zipped and added to the website file directory, updating the **file path** constant within the server javaScript where necessary to match the location of the packaged program.

7.2 - REPORTING BUGS:

If any bugs are found within the SewAnalyst application, they can be reported via the GitHub "issues" feature. The reported information, including details of where the errors occurred within the program, should be as specific as possible. Using this, the maintenance team can attempt to fix any errors within the software and deploy patches and updates.



7.3 - ADDING NEW FUNCTIONALITY:

New functionality may be added to the software through the pre-defined flask routing system. Flask routing works by associating a function to a URL, meaning by navigating different requests, different backend functionality can be called automatically.

The code for the functionality of the proposed new feature should be added to the "library.py" file. This file is already imported to main and all functions withing should be immediately accessible in "main.py".

7.4 - MODIFYING ANALYSIS:

Should the software development team wish to alter in process by which statistics are created or implement new statistics, library.py can be modified to accommodate this. At present, there are 3 functions which are used in the creation of statistics: calculateDifference, fftThreshold and questionThree. The details on how these functions operate can be found in the **Application Implementation Details** section of this user manual. To interface with the report generation process, it is important that these functions return True if there is a potential violation of guidelines and False if they do not. They may also return a severity of violation which may be classified as "none", "mild", or "severe".

To implement new statistics, new functions can be created to process their creation. They must return the correct values and store them within the "stats" dictionary located in the main function. questionThree may need to be altered if the new statistics influence the answer to Q3.

Please Note:

The product owner should stay up to date on current guidelines from the EA and alter the data analysis process if they change.

7.5 – MONITORING REPORTS

The software team may wish to edit the JSON file storing the report data. To do this, they would need to access the flaggedReports.json file. This is located within "./website-frontend/files/flaggedReports.json".

Once the file has been opened it will look something like this:

1 The software engineer may wish to change the report name, as some reports may be uploaded with the same name / have inappropriate names.

2 If certain reasons that a report have been flagged appear invalid, the engineer can remove the data objects. Removing every object within the array of the "review" attribute for every invalid reason. For example, the highlighted box should be completely removed.

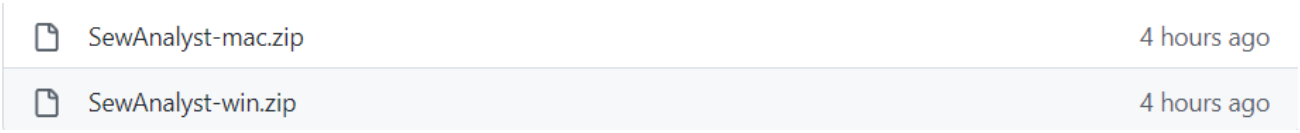
3 The 'reportCounter' attribute should then be updated to correlate to the number of "reviews". This should only need changing if the software developer manually removes or adds reviews as outlined above in part 2.

To remove an entire report, the whole section highlighted in green should be removed. Along with this, in the './website-frontend/files/reports' folder the pdf file corresponding to the name of the report that has just been deleted should also be deleted from that folder.

7.6 - SOFTWARE UPDATE DEPLOYMENT

If the software is updated, the new builds of the program will need to be added to the website. Upon building the updated applications for each distribution, the developer should navigate to the following directory: './website-frontend/files/program'.

The old version firstly needs removing from this directory using 'shift + delete' to permanently delete the .zip file. The newly packaged software, in accordance with the **Building the Software** section must then be zipped and added to this directory. The directory should contain the 'SewAnalyst-mac.zip' and 'Sewanalyst-win.zip' files before being deployed.



8 – FUTURE FEATURES

Listed below are a collection of future planned features for our website and software.

8.1 - WEBSITE-SPECIFIC FUTURE FEATURES

a) Admin page for form validation and removal

A content moderation team should be assigned to monitor uploaded reports which have been flagged as suspicious. Currently, only a developer can edit the website front-end code to manually remove both uploaded reports and reasons for flagging a report. An admin page instead of simply automating report moderation (through the code) is a better method of doing this as moderation automation is vulnerable to false flagging attacks (see 1.b), which might cause issues to our meta-analysis of all the data (see 1.d). Therefore, potential report moderation features on an admin page would include as follows:

- The ability to easily remove a report if it is in fact incorrect or suspicious.
- The ability to verify a report and make it "un-flaggable" if it has been checked by the admins (to avoid repeatedly checking the same reports), and the ability to provide a justification on a pop-up for why you can no longer flag this certain report.
- The ability to easily remove incorrect or potentially malicious flagging reports of any uploaded report, potentially in bulk (see 1.b).

Such an admin page could be created using a backend database or platform service such as Microsoft Azure.

b) Protection against false flagging attacks

We have not accounted for potential malicious flagging attacks on valid reports, which might skew public opinion on the validity of our software if we cannot mediate these attacks fast enough. To avoid this issue, we could add future features such as adding a cut-off point for numbers of time you can flag a report (realistically once per session) or could add the ability to notify the content moderation team (see 1.a) if there has been unnaturally high rates of flagging in one day, who could then mass delete complaints from a report.

c) Report filtering

The current system works best for a low number of uploaded reports, as there is no current implementation to filter or query reports. Therefore, some sort of filtering system and report query process would be a very useful future feature for administration. This would be implemented via a backend database and query language such as SQL, as defined below in section **8.1e).**

d) Automatic upload of report from software

Manual report upload is currently required for report upload due to the security risks involved with allowing API requests from locally hosted external applications to our website. In order to implement automatic uploads, there are one of two methods:

1. Add multiple security measures and validation to the API of both the software and website to allow for safe transfer of files through requests made by Flask.
2. Integrate the software into the website as a web-application. This is expanded in section **8.2b)**.

e) Database for data storage

Having a robust database built into the backend of the website would allow for the management and storage of reports and their associated data, giving way to the method of meta-analysis defined in our initial requirements and expanded upon below (**8.1f**). It would also allow for the querying and filtering of reports by administration teams and end-users, making the accessibility of the reports far easier.

f) Meta-analysis of all data

Meta analysis is a feature in which the calculated data and statistics for individual STWs are cross-compared and validated in order to highlight common discrepancies and trends. Trends could be in reference to either locational based trends, or trends pertaining to discrepancies and which methods of circumvention STWs commonly used for prescribed EA guidance.

8.2 - SOFTWARE-SPECIFIC FUTURE FEATURES

a) Robust Data Validation and Cleaning

At the moment, we ask users to provide data as a csv file containing only the required attributes in a specific order. This might be time-consuming and frustrating for some users and might decrease the likelihood of them finishing the uploading process for report generation. As we want as many reports to be generated as possible in order to analyse the general picture (meta-analysis), a change to this method of file upload would be necessary. Increasing robust data validation methods such as automatically filtering the dataset to identify the necessary attributes and expanding the cleaning capabilities of our software would allow a user to easily upload a bigger, less formatted dataset.

b) Website Integration

Currently, the **SewAnalyst** software runs as a standalone downloadable software, however in the future there are plans for integration into the website. This would allow for seamless report generation upload, as well as lowering the system requirements. It would also allow, in conjunction with the implementation of a website database, the ability to provide dynamic meta-analysis based on the data set of currently uploaded reports.

9 – BIBLIOGRAPHY

1. Cloud computing services: Microsoft Azure (no date) Cloud Computing Services | Microsoft Azure. Available at: https://azure.microsoft.com/en-gb/ (Accessed: March 16, 2023).
2. Cloud computing services - amazon web services (AWS) (no date). Available at: https://aws.amazon.com/ (Accessed: March 16, 2023).
3. pyfpdf.readthedocs.io. (August 15 2012). PyFPDF. [online] Available at: https://pyfpdf.readthedocs.io/en/latest
4. Readthedocs.io. (2011). Pillow — Pillow (PIL Fork) 6.2.1 documentation. [online] Available at: https://pillow.readthedocs.io/en/stable/.
5. pyfpdf.readthedocs.io. (2012). Reference manual - PyFPDF. [online] Available at: https://pyfpdf.readthedocs.io/en/latest/ReferenceManual/index.html#pyfpdf-reference-manual