Page Speed Analytics Application - Capstone Project Paper

Lillian C. Arguelles

Nova Southeastern University

# 

[**1 Introduction**](#_olhu44capel) **4**

[1.1 Overview](#_6k0nwth2cfse) 4

[1.2 Scope](#_u1wcood54wlp) 5

[1.3 Definitions and Acronyms](#_xztwd1nb8u85) 5

[**2 Design Overview**](#_rzsnjlyywvl) **6**

[2.1 Data Design](#_w0d0de79li5l) 6

[2.1.1 Overview](#_loepkxq1nlhj) 6

[2.1.2 Page Speed Entity class object](#_pnb12ls6v912) 6

[2.1.3 External CSV Input File](#_7suwkxcwyhia) 6

[2.2 Architectural Design](#_xzzdyfst03re) 7

[2.2.1 Structure of the Page Speed Analytics Application](#_xgm0u8u5blph) 7

[Figure 1: Structure and class interactions in Page Speed Analytics Application](#_1nep6pt93s8f) 7

[2.3 Class and Object Design/Modules](#_gdoxlilaj0p7) 7

[2.3.1 API Key Storage Helper Class ( APIKeyStorage )](#_hccddfa9slk5) 8

[2.3.2 HTML Report Viewer Class ( ReportHTMLViewer )](#_mlsk2jgbati9) 8

[2.3.3 Page Speed Entity Class ( PageSpeedEntity )](#_z48ix47e1rhy) 9

[2.3.4 Page Speed Calculations Class ( PageSpeedCalculations )](#_btbttdqjmwlc) 9

[2.3.5 Page Speed Report Class ( PageSpeedReport )](#_kew8uyevikiv) 10

[2.3.5.A CSV Output File Naming Convention](#_yadginxmv5c3) 12

[2.3.6 Page Speed Service Class ( PageSpeedService )](#_tawmwoqhjjhz) 13

[2.3.7 Program Main Class ( Program )](#_jhys9wvktpsx) 14

[2.3.8 Read Write CSV Class ( ReadWriteCSV )](#_9x63mojqzv77) 14

[2.3.9 Run From CSV Class ( RunFromCSV )](#_wl6tphnd1n34) 15

[2.3.10 Run From Text Class ( RunFromTxt )](#_vwew9oah9nhh) 16

[2.3.11 Runner Process Class ( RunnerProgress )](#_vf2oxc3g0qrm) 18

[2.3.12 Welcome Class ( Welcome )](#_ggwgjhn4fzf8) 20

[2.4 Interface Design](#_scoo35vdlq6o) 21

[2.4.1 Welcome GUI (Welcome)](#_6uubqoddy1yv) 21

[Figure 2: Welcome GUI Interface, using the Welcome Windows Form](#_vzgat1p3cvjx) 21

[Figure 3: Welcome Interface Button Process Flow](#_1gojsfuexkjr) 22

[2.4.2 Run From CSV GUI (RunFromCSV)](#_qzemrftuwtcm) 23

[Figure 4: Run From CSV GUI Interface, using the RunFromCSV Windows Form](#_jejo65mm7k94) 23

[2.4.3 Run From Text GUI (RunFromTxt)](#_dwbk2jwp2e1t) 24

[Figure 5: Run From Text GUI Interface, using the RunFromTxt Windows Form](#_w5csiyn4xmwk) 24

[2.4.4 Runner Progress GUI (RunnerProgress)](#_ufpjlckak2ur) 24

[Figure 6: Runner Progress GUI Interface, using the RunnerProgress Windows Form](#_b21uhx59k3pc) 25

[2.4.5 HTML Report View GUI (ReportHTMLViewer)](#_ncr9mvbbekjz) 25

[Figure 7: HTML Report View GUI Interface, using the ReportHTMLViewer Windows Form](#_1ywlijfqmkjq) 26

[**References**](#_ie0ju6idepex) **26**

# 

# 

# 

# 1 Introduction

## 1.1 Overview

In this past January, Google announced on their Official Google Webmaster Central Blog that starting in July of 2018, Page Speed will be one of the factors thats start to impact online ecommerce sites, blogs and other online businesses. Primarily this will be impacting their mobile rankings in Google Searches. While this makes sense from the general consumers perspective, from the business and site owners perspective, being impacted by this could cause an impact to their revenue. As more and more users are searching and buying products from their mobile devices, being at the top of the search rankings has become vital.

A standard recommendation to many site owners is to track and log the statistics on their website when they are undergoing changes due to making updates and improvements to their site. Currently, to do this site owners have one of two choices: manually log the statistics or pay a company (GTMetrix, Pingdom, KeyCDN, etc) to do so. Running and logging these tests on more than one url was a time-consuming and complicated process. Due to this, many site owners either chose to pay a company a monthly fee, to only check sporadically one or two urls, or to just not track the information at all. This is where the Page Speed Analytics Application can help fill in this gap.

Page Speed Analytics Application (PSAA) is a C# based application that assists users in running page speed tests on either a single url or a group of urls. It creates a report based on these urls, that can be saved as a CSV file, viewed as an html file, or both. Users only need to either load in a CSV file containing the urls to run the reports on or type in the Urls into the provided text boxes. The user then does not need to do anything else except hit their preferred report method, and click submit. PSAA will take care of everything else.

## 1.2 Scope

Currently PSAA does have a few limitations and constraints:

* Users do need to be sure the url is a valid url, if it is not, PSAA will simply not run the service for that url.
* The user needs to launch the C# Windows Form Application in Visual Studio and click the Run button for it to run.
* The following constraints are also found in PSAA due to the constraints on the Google Page Speed Insight API:

Number of Total Queries per Day, per API Key: 25,000 queries

Number of Total Queries per 100 seconds, per API Key: 1,000 queries

Number of Total Queries per 100 seconds, per user, per API Key: 100 queries

Currently due to the parameters set already in PSAA, this means the total number of urls that can be run per 100 seconds is 50 (as it is two queries per each url).

* The API Key used (and hardcoded in PSAA) is linked to the designer’s own personal Google Account. The steps that a user would need to take to create their own API Key and enter it into the project.

## 1.3 Definitions and Acronyms

PSAA Page Speed Analytics Application

PSI Google Page Speed Insights (both an API and a website)

Speed Score The numeric integer score (0 - 100) that the Google Page Speed Insights gives based on a number of factors. While many of the factors are known, the weight of or value assigned to the factors is not known.

# 2 Design Overview

## 2.1 Data Design

### 2.1.1 Overview

There is one main data model that is used in the Page Speed Analytics Application: the Page Speed Entity class object (PageSpeedEntity.cs). There is also two optional external file structures used in PSAA, a CSV file containing urls to input into PSAA and run the reports on, and the html template file that is used in the html report viewing process.

### 2.1.2 Page Speed Entity class object

The PageSpeedEntity object is the primary storage object for PSAA. Once a Page Speed API request is made the returned result is in the form of a custom class object. This is then stored as a PageSpeedEntity object. The original structure of the object as a custom object is preserved and stored in the the result property in the PageSpeedEntity object. This allows for future modifications and additions to be made to the report classes without needing to change the creation and storage procedures.

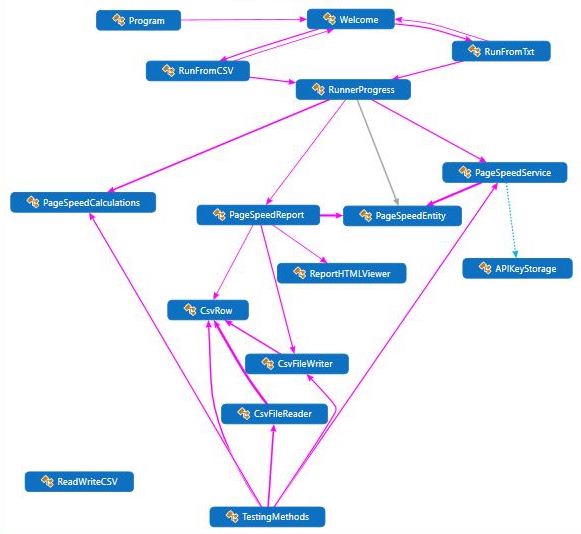
*Note: See Section 2.3 for more detailed information about the PageSpeedEntity Class.*

### 2.1.3 External CSV Input File

Included with the application is an example CSV file for a user to see how to format the CSV file for better processing. It is simple a single column of urls, with one url per row.

## 2.2 Architectural Design

### 2.2.1 Structure of the Page Speed Analytics Application



##### Figure 1: Structure and class interactions in Page Speed Analytics Application

## 2.3 Class and Object Design/Modules

The Page Speed Analytics Application utilizes 12 classes in total for it’s main functioning and one class that is used for testing purposes. Of the 12 main classes, 5 are interfaces (one of which is additionally a main processing class), three helper classes, one data model class and three processing classes.

### 2.3.1 API Key Storage Helper Class ( APIKeyStorage )

The API Key Storage class is a helper class that stores the application name and api key as a string for usage in the Page Speed Service class during the creation of the Page Speed Online Service. This allows for future changes and improvements to be made without needing to make changes to the service class.

The API Key Storage class does not contain any methods or functions.

The global variables are listed below:

|  |  |
| --- | --- |
| **Variable** | **Description** |
| public string API\_KEY | Text string of api key for Google API |
| public string APP\_NAME | Text name of application for Google API |

### 

### 2.3.2 HTML Report Viewer Class ( ReportHTMLViewer )

The HTML Report Viewer Class is a GUI Windows Form that contains a WebBrowser Component. The component during initialization and loading, displays an html file and then writes the string (htmlTextString) that was passed in as a parameter during creation to the html document.

|  |  |
| --- | --- |
| **Variable** | **Description** |
| public string htmlTextString | string of html text to show |

|  |  |
| --- | --- |
| ReportHTMLViewer | *Descrip*:Constructor. Sets the global htmlTextString variable.Initializes the components. |
| *Params*:htmlTextString (string) |
| *Returns*:None |
| ReportHTMLViewer\_Load | *Descrip*:On load of the ReportHTMLViewer Form, this function loads the template html file (located within the project's own directory) into the WebBrowser component, displays the page and writes the value of the htmlTextString variable to the html page. |
| *Params*:sender (Object), e (EventArgs) |
| *Returns*:None |

*Note: See Section 2.4.5 for information about the interface design in the ReportHTMLViewer Class.*

### 2.3.3 Page Speed Entity Class ( PageSpeedEntity )

The Page Speed Entity class is the main data model storage object class for the Page Speed Analytics Application. This custom object class uses five properties with get and set accessors. The five properties are listed below.

|  |  |
| --- | --- |
| string Base\_url | Url that was put in the request |
| Google.Apis.Pagespeedonline.v4.Data.PagespeedApiPagespeedResponseV4 Result | Response returned from the Google PageSpeed v4 API. Stored in the same format as it was received for easier storage and access. |
| int Score | Page Speed Score ( 0 - 100 ) |
| DateTime Date | Date of request / result |
| int Strategy | The integer that indicates what strategy was used for the request. 0 for Desktop; 1 for Mobile. |

### 2.3.4 Page Speed Calculations Class ( PageSpeedCalculations )

The Page Speed Calculations class is a helper class for the Page Speed Analytics Application. It contains four methods for its’ three primary usages.

The three usages are: parsing a CSV into a list of strings (ParseCSV) which is utilized by RunnerProgress; converting a given string containing a report method, and converting it into an integer report method (ConvertReportMethod) which is utilized by RunnerProgress. The final usage is checking a list of urls to see if they are valid, and returning only the valid url strings. This uses two of the methods, CheckIfUrlsIsValid and CheckIfUrlIsValid. It was designed this way for ease of use. Currently, in the class RunnerProgress, prior to running the list through the PageSpeedService process, the given list of urls (either by CSV or Text Input) is run through this validation check. Non-valid urls are removed and valid ones are allowed to be added to the list being returned.

|  |  |
| --- | --- |
| ParseCSV | *Descrip*: Parse a CSV file into a list of strings |
| *Params*: path (string) |
| *Returns*: List<string> |
| CheckIfUrlsIsValid | *Descrip*: Checks a given string list if the urls are valid and  Returns only the valid urls. |
| *Params*: urls ( List<string> ) |
| *Returns*: List<string> |
| CheckIfUrlIsValid | *Descrip*: Checks a given string to see if it is a valid url.  Returns TRUE if it is valid, else returns FALSE. |
| *Params*: strURL ( string ) |
| *Returns*: boolean result |
| ConvertReportMethod | *Descrip*: Converts given string into an int report method.  Returns the integer report method.  0 = CSV, 1 = HTML, 2 = Both |
| *Params*: s ( string ) |
| *Returns*: int |

### 2.3.5 Page Speed Report Class ( PageSpeedReport )

The Page Speed Report Class runs the report creation and display procedures. Upon the classes initialization, the function CreateReports is called.

First the CreateReports function, creates the table of information used in the report. creates the header array (the strings that are for populating the header row in the final table, then it sets the strings from the header row into the main storage multi dimensional array for the main table (the string[,] rows variable). It then populates the rest of the “rows” in the variable rows with the information needed from the stored responses. This was passed in as a parameter as two lists of PageSpeedEntity objects called deskResults, for the desktop results, and mobileResults, for the mobile results. Once done it uses the passed in integer method to determine the report creation type; 0 for CSV only report, 1 for HTML report only or 2 for both the CSV and HTML reports. For CSV or Both it calls the function CreateCSVReport, and for HTML or Both it calls CreateHtmlReport.

For the CSV Report creation, CreateCSVReport creates a csv file that contains the passed in table (multi-dimensional string array). It is saved with the filename: “PageSpeedCSVReportMMDDYYYY-hhmmtt.csv”. The naming convention is explained in CSV Output File Naming Convention section 2.3.5.A below.

|  |  |
| --- | --- |
| CreateCSVReport | *Descrip*: Takes in a multi-dimensional string array and creates a CSV file to store it. Saves file to user's Downloads folder. |
| *Params*: data ( string[,] ) |
| *Returns*: None |
| CreateHtmlPage | *Descrip*: Create text for a HTML file. |
| *Params*: data ( string[,] ) |
| *Returns*: String containing text of a newly created HTML page. |
| CreateHTMLReport | *Descrip*: Takes in a multi-dimensional string array and creates an html file to show it. |
| *Params*: data ( string[,] ) |
| *Returns*: None |
| CreateHTMLTable | *Descrip*: Create text for a HTML table. |
| *Params*: data ( string[,] ) |
| *Returns*: String containing text of a newly created HTML Table. |
| CreateReports | *Descrip*: Runs create report processes |
| *Params*: urls ( List<string> ), deskResults ( List<PageSpeedEntity> ), mobileResults ( List<PageSpeedEntity> ), method ( int ) |
| *Returns*: None |
| GetHeaderString | *Descrip*: Create the string array for the header |
| *Params*: None |
| *Returns*: string array containing the header strings for CSV file |

#### 2.3.5.A CSV Output File Naming Convention

The output CSV file name is made up of two parts (the prefix and suffix) combined with the file extension. The prefix is “PageSpeedCSVReport\_” and the extension “.csv”. The suffix is a formatted DateTime string. It formats the current date and time (at the time of file creation) to the format of MMddyyyy-hhmmtt, where MM for the month integer (05 for May), dd for the day (01 for the first), yyyy for the year (2018 for the year 2018), hh for the hour in the 12-hour format (09 for 9 AM), mm for the minutes (00 for 0 minutes) and tt for the AM/PM designator (AM for the AM morning designator). For example, May 1st, 2018 9:00 AM would be 05012018-0900AM. The chart below show this with an example.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Prefix** | **Suffix** | **Ext.** | **Put-Together** |
| **Format** | PageSpeedCSVReport\_ | MMddyyyy-hhmmtt | .csv | PageSpeedCSVReport\_MMddyyyy-hhmmtt.csv |
| **Example** | PageSpeedCSVReport\_ | 05012018-0900AM | .csv | PageSpeedCSVReport\_05012018-0900AM.csv |

### 2.3.6 Page Speed Service Class ( PageSpeedService )

Page Speed Service is the class that controls the interactions with the Google PageSpeed API. It utilizes the Google.Apis.Pagespeedonline.v4 package to send requests to the api.

To use this class, the function RunProcessUrl is called and a string with the url to use is passed in. A List of size two of PageSpeedEntity objects is initialized. An instance of the PageSpeedService is created, and the RunPageSpeedServiceProcess Task is called, with the parameters for desktop, the index of the list element and the url string passed in. The result from the task is then accessed by RunProcessUrl and passed into the CreatePageSpeedEntity function to create a PageSpeedEntity object along with the url string and the integer indicator for setting it as a desktop result. The returned PageSpeedEntity object is added to the list of PageSpeedEntity objects. This process is repeated to get the mobile results. After both requests are sent, responses received and stored as PageSpeedEntity objects in a list, the list is returned to the class that called the RunProcessUrl function.

The RunPageSpeedServiceProcess Task creates the service, Pagespeedonlineservice using the BaseClientService (classes from the Google.Apis.Pagespeedonline.v4 package), sets the strategy, executes the request, gets and returns the results.

CreatePageSpeedEntity takes in the url string, the result received from RunPageSpeedServiceProcess in RunProcessUrl, and the strategy integer. It creates the PageSpeedEntity object and then returns it.

|  |  |
| --- | --- |
| CreatePageSpeedEntity | *Descrip*: Create the page speed entity object. |
| *Params*: strURL ( string ), result ( Google.Apis.Pagespeedonline.v4.Data.PagespeedApiPagespeedResponseV4 ), strategy ( int ) |
| Returns: PageSpeedEntity object that was created |
| RunPageSpeedServiceProcess | *Descrip*: Task that makes the request to the api, gets the response and then returns the response. |
| *Params*: strategy ( int ), url ( string ), index ( int ) |
| *Returns*: Task<Google.Apis.Pagespeedonline.v4.Data.PagespeedApiPagespeedResponseV4> |
| RunProcessUrl | *Descrip*: Runs request on url and returns the response |
| *Params*: url ( string ) |
| *Returns*: List of PageSpeedEntity objects with results |

### 2.3.7 Program Main Class ( Program )

The Program class is the starting point of the Page Speed Analytics Application, in the Main function, it starts up the application and instantiates the Welcome form class.

|  |  |
| --- | --- |
| Main | *Descrip*: Main entry point for the application.  Starts up the Welcome form. |
| *Params:* None |
| *Returns:* None |

### 2.3.8 Read Write CSV Class ( ReadWriteCSV )

The ReadWriteCSV class is utilized as a helper class in the Page Speed Analytics Application. It is not originally created by the developer but rather by Jonathon Wood on CodeProject.com as part of an online tutorial progam. See the reference section for the reference information to this.

The class is made up of three inner classes: CsvRow, CsvFileWriter and CsvFileReader. This class creates a custom StreamWriter implementation to make it easier to read and write from CSV files.

### 2.3.9 Run From CSV Class ( RunFromCSV )

The Run From CSV Class is one of the main GUI windows form classes used in the Page Speed Analytics Application. It contains a OpenFileDialog component that allows the user to browse their computer and select a CSV file for the application to use. Once done the user chooses a report method (CSV, HTML or Both) and clicks the submit button. The Submit button will send the file information (file name and path) and report method type to the RunnerProgress Class (Section 2.3.11) to process the CSV and create the result reports.

|  |  |
| --- | --- |
| **Variable** | **Description** |
| public string returnMethod | Report method, set by user in the form. |
| public string fileNameToSubmit | File Name of the input CSV file. |
| public string filePathToSubmit | File Path of the input CSV file. |

|  |  |
| --- | --- |
| ChooseFileButton\_Click | *Descrip*: Starts procedure from the OpenFileDialog Component to allow the user to select a file. Gets the name and path from the file selection, stores it in the global variables and displays it in the applicable labels. |
| *Params*: sender (Object), e (EventArgs) |
| *Returns*: None |
| CSVGoButton\_Click | *Descrip*: Starts procedure when Submit Button is clicked. Starts RunnerProcess, passes in file information and gives applicable messages where needed. |
| *Params*: sender (Object), e (EventArgs) |
| *Returns*: None |
| HelpButton\_Click | *Descrip*: Shows the Help message when Help Button is clicked. |
| *Params*: sender (Object), e (EventArgs) |
| *Returns*: None |
| radioButton1\_CheckedChanged | *Descrip*: Set returnMethod to CSV. |
| *Params*: sender (Object), e (EventArgs) |
| *Returns*: None |
| radioButton2\_CheckedChanged | *Descrip*: Set returnMethod to HTML. |
| *Params*: sender (Object), e (EventArgs) |
| *Returns*: None |
| radioButton3\_CheckedChanged | *Descrip*: Set returnMethod to Both. |
| *Params*: sender (Object), e (EventArgs) |
| *Returns*: None |
| RunFromCSV | *Descrip*: Constructor |
| *Params*: None |
| *Returns*: None |
| RunFromCSV\_FormClosed | *Descrip*: Open Welcome Form when current form is closed. |
| *Params*: sender (Object), e (EventArgs) |
| Returns: None |
| RunFromCSVCancelButton\_Click | *Descrip*: Closes form when Cancel Button is clicked. |
| *Params*: sender (Object), e (EventArgs) |
| *Returns*: None |
| ShowMessage | *Descrip*: Uses MessageBox to show user a message. |
| *Params*: message ( string ), caption ( string ), btns (  MessageBoxButtons ), icon ( MessageBoxIcon ) |
| *Returns*: None |
| UpdateFileNameLabel | *Descrip*: Updates the the CSVFileNameLabel |
| *Params*: f ( string ), p ( string ) |
| *Returns*: None |

*Note: See Section 2.4.2 for information about the interface design in the RunFromCSV Class.*

### 2.3.10 Run From Text Class ( RunFromTxt )

The Run From Text Class is one of the main GUI windows form classes used in the Page Speed Analytics Application. It contains a five text boxes that the user can use to fill in with urls for the application to use. The user can fill in one, a few or all of the textboxes. Once done the user chooses a report method (CSV, HTML or Both) and clicks the submit button. The Submit button will send the url strings and report method type to the RunnerProgress Class (Section 2.3.11) to process the urls and create the result reports.

|  |  |
| --- | --- |
| **Variable** | **Description** |
| public string returnMethod | Report method, set by user in the form. |
| public string DEFAULT\_RETURN\_METHOD | Default report method. |

|  |  |
| --- | --- |
| CancelButton\_Click | *Descrip*: Closes form when Cancel Button is clicked. |
| *Params*: sender (Object), e (EventArgs) |
| *Returns*: None |
| ClearTextBoxes | *Descrip*: Clears the textboxes on the form |
| *Params*: None |
| *Returns*: None |
| GetUrlsFromTextBoxes | *Descrip*: Gets url strings from the textboxes |
| *Params*: None |
| *Returns*: None |
| HelpButton\_Click | *Descrip*: Shows the Help message when Help Button is clicked. |
| *Params*: sender (Object), e (EventArgs) |
| *Returns*: None |
| radioButton1\_CheckedChanged | *Descrip*: Set returnMethod to CSV. |
| *Params*: sender (Object), e (EventArgs) |
| *Returns*: None |
| radioButton2\_CheckedChanged | *Descrip*: Set returnMethod to HTML. |
| *Params*: sender (Object), e (EventArgs) |
| *Returns*: None |
| radioButton3\_CheckedChanged | *Descrip*: Set returnMethod to Both. |
| *Params*: sender (Object), e (EventArgs) |
| *Returns*: None |
| RunFromTxt | *Descrip*: Constructor |
| *Params*: None |
| *Returns*: None |
| RunFromTxt\_FormClosed | *Descrip*: Open Welcome Form when current form is closed. |
| *Params*: sender (Object), e (EventArgs) |
| *Returns*: None |
| ShowMessage | *Descrip*: Uses MessageBox to show user a message. |
| *Params*: message ( string ), caption ( string ), btns (  MessageBoxButtons ), icon ( MessageBoxIcon ) |
| *Returns*: None |
| SubmitButton\_Click | *Descrip*: Starts procedure when Submit Button is clicked. Gets the urls, starts RunnerProcess, gives applicable messages where needed. |
| *Params*: sender (Object), e (EventArgs) |
| *Returns*: None |

*Note: See Section 2.4.3 for information about the interface design in the RunFromTxt Class.*

### 2.3.11 Runner Process Class ( RunnerProgress )

The Runner Progress Class is one of the main GUI windows form classes used in the Page Speed Analytics Application. While it is a windows form, the user is not able to interact with the form. Runner Progress is from the user’s perspective simply a progress bar and status update GUI. The form uses a ProgressBar component that increments to show it’s progress through the procedures being run in the background.

From the computer perspective, Runner Progress is the main director class. It can be instantiated and called from either of the two main interactive GUI forms (RunFromCSV or RunFromTxt) and once instantiated it controls which procedures are run and classes run for the majority of the processing.

From RunFromCSV it is started in the ProcessFromCSV method, where the file information and report method is passed in. From there, a list of strings containing urls is created using the ParseCSV method in PageSpeedCalculations, the ProgressBar is incremented and the string list is then passed into the Process method to finish processing like the RunFromTxt class,

From RunFromTxt (and the ProcessFromCSV via the RunFromCSV class) the string list and report method type is passed in and the status string is updated for the user to see, letting them know the processing has started. The list of urls is then run through the validation process using CheckIfUrlsValid from PageSpeedCalculations, and invalid urls are removed from the list. The report method type is then converted into an integer for later use. Then for each of the url strings, the PageSpeedService class is instantiated, called and results returned from in the form of two PageSpeedEntity objects in a PageSpeedEntity list per url (one for desktop, one for mobile).

The status string is updated to let the user know when this process is finished and the report creation process has been started. The PageSpeedReport class is then called with the CreateReports method, the string urls, PageSpeedEntity objects and converted report method is then passed in. Once the CreateReports method has finished the report creation procedure the RunnerProgress GUI is closed.

|  |  |
| --- | --- |
| **Variable** | **Description** |
| public bool runCSV | Bool value of whether this was started from CSV form. |
| public int DEFAULT\_STEP | Default integer amount of step for the ProgressBar. |

### 

|  |  |
| --- | --- |
| IncreaseBar | *Descrip*: Increases the amount shown on the ProgressBar Component |
| *Params*: amt ( int ) |
| *Returns*: None |
| Process | *Descrip*: Processes a string list of urls to return a report method matching the type in the string report method parameter |
| *Params*: urls ( List<string> ), reportMethod ( string ) |
| *Returns*: bool value of whether it ran or not |
| ProcessFromCSV | *Descrip*: Starts the process with a csv file. |
| *Params*: fileName ( string ), filePath ( string ), reportMethod ( string ) |
| *Returns*: bool value of whether it ran or not |
| RunnerProgress | *Descrip*: Constructor. Starts up the form. |
| *Params*: None |
| *Returns*: None |
| RunnerProgress\_Load | *Descrip*: Runs at the forms load time. Sets up the status label and ProgressBar. |
| *Params*: sender (Object), e (EventArgs) |
| *Returns*: None |
| UpdateStatus | *Descrip*: Updates the status label with a passed in string (s) and increases the ProgressBar by the passed in integer amount (amt). |
| *Params*: amt ( int ), s ( string ) |
| *Returns*: None |
| UpdateStatusLabel | *Descrip*: Updates the status label with a passed in string (s). |
| *Params*: s ( string ) |
| *Returns*: None |

*Note: See Section 2.4.4 for information about the interface design in the RunnerProgress Class.*

### 2.3.12 Welcome Class ( Welcome )

The Welcome class is the first form class that will be displayed for the user. From here users choose to run the process by text (typing in the urls) or by csv (uploading a csv file to the procedure). Once they make a choice, the welcome form will instantiate and display the applicable form.

|  |  |
| --- | --- |
| **Variable** | **Description** |
| public RunFromTxt runFromTxtForm | Form to use for entering in text urls. |
| public RunFromCSV runFromCSVForm | Form to use for entering using a csv file. |

|  |  |
| --- | --- |
| Welcome | *Descrip*: Constructor. Starts up the Welcome Form. |
| *Params*: None |
| *Returns*: None |
| EnterURLButton\_Click | *Descrip*: Starts the Run From Text Form Window |
| *Params*: sender (Object), e (EventArgs) |
| *Returns*: None |
| EnterCSVButton\_Click | *Descrip*: Starts the Run From CSV Form Window |
| *Params*: sender (Object), e (EventArgs) |
| *Returns*: None |
| HelpButton\_Click | *Descrip*: Shows the Help Message |
| *Params*: sender (Object), e (EventArgs) |
| *Returns*: None |
| QuitButton\_Click | *Descrip*: Closes the application |
| *Params*: sender (Object), e (EventArgs) |
| *Returns*: None |
| RunFromCSV\_FormClosed | *Descrip*: Opens the Welcome Form when the  Run From CSV Form has closed. |
| *Params*: sender (Object), e (EventArgs) |
| *Returns*: None |
| RunFromTxt\_FormClosed | *Descrip*: Opens the Welcome Form when the Run From Text Form has closed. |
| *Params*: sender (Object), e (EventArgs) |
| *Returns*: None |

*Note: See Section 2.4.1 for information about the interface design in the Welcome Class.*

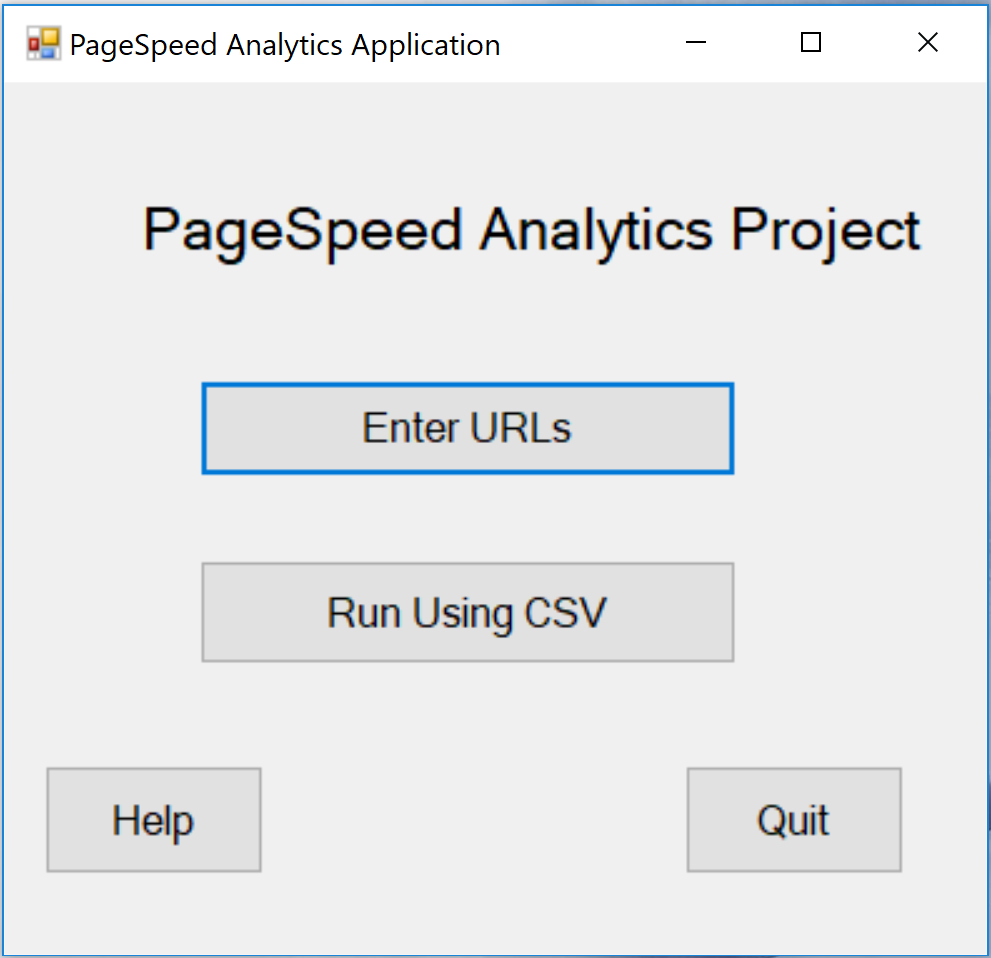
## 2.4 Interface Design

HCI, External Interfaces, internal interfaces

The Page Speed Analytics Application contains five main interface screens that were created as Windows Forms in Visual Studio 2017.

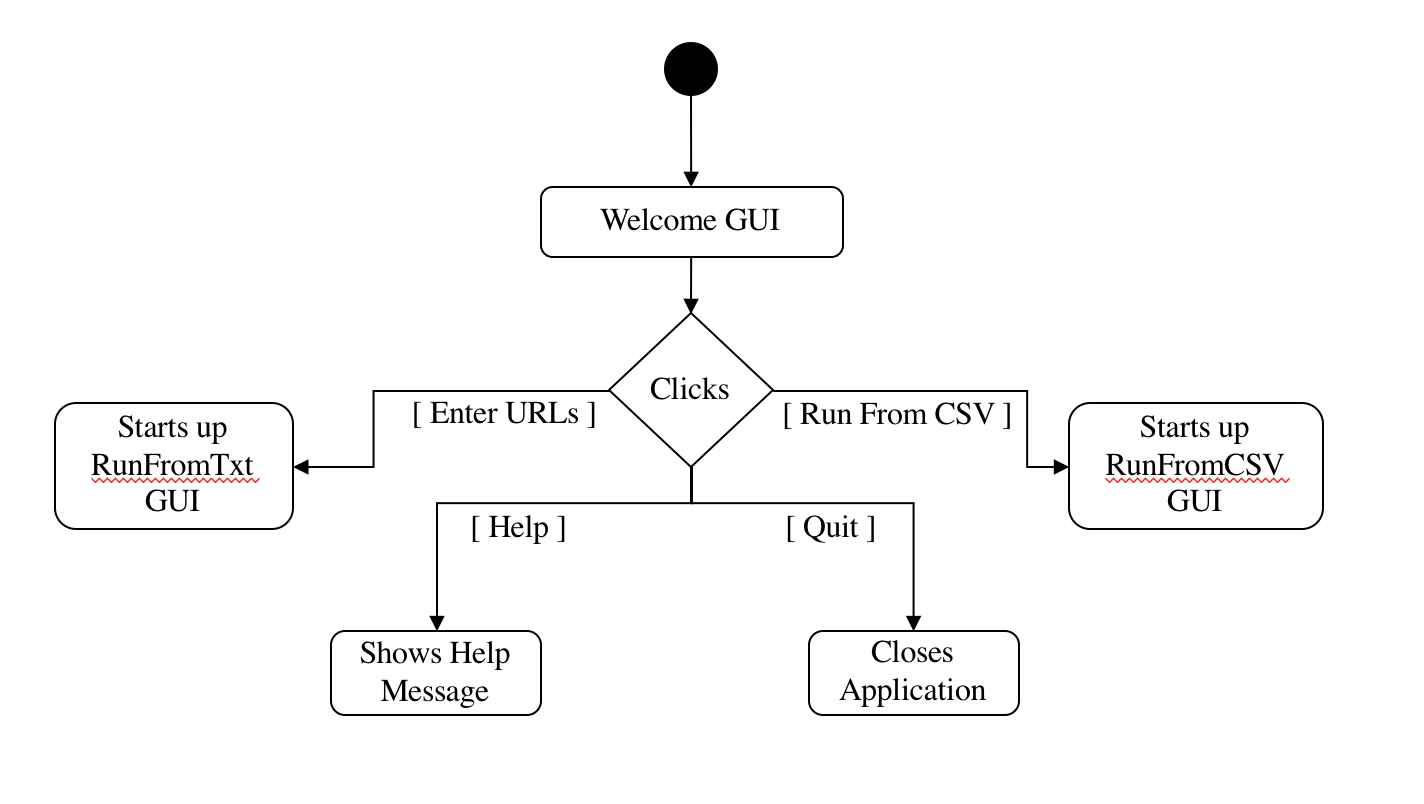
#### 2.4.1 Welcome GUI (Welcome)

The First screen a user sees when PSAA starts up is the welcome screen. From here the user can either click one of four buttons: the *Enter URLs* button, the *Run Using CSV* button, the *Help* button or the *Quit* button.



##### Figure 2: Welcome GUI Interface, using the Welcome Windows Form

Each of these launches a specific functioning as shown in Figure 3 below.



##### Figure 3: Welcome Interface Button Process Flow

The Help process utilizes a Windows Form MessageBox component that displays a string message explaining to the user what they are able to do.

The Quit process when clicked tells the application to start the shut-down procedure. This entails closing the Welcome GUI display and exiting the application. This procedure can also be prompted by clicking the *X* button from the top right of the toolbar.

The Enter URLs Process initializes the RunFromTxt Windows GUI Form and displays it. Additionally the process hides the Welcome GUI from the user’s view allowing it to be re-displayed later on should the user exit from the RunFromTxt GUI form.

The Run Using CSV Process initializes the RunFromCSV Windows GUI Form and displays it. Additionally the process hides the Welcome GUI from the user’s view allowing it to be re-displayed later on should the user exit from the RunFromCSV GUI form.

*See Section 2.3.12 for the class and method information for the Welcome class.*

#### 2.4.2 Run From CSV GUI (RunFromCSV)

When the user prompts the Run From CSV Process, the RunFromCSV GUI is displayed. This is the first of two methods that the user can choose from to use the PSAA application.

The Run From Text GUI utilizes three buttons: a *Cancel* button, a *Help* button and a *Submit* button. Additionally, for selecting the report method (the method in which the user can request to get a report of the information from the given urls), the RunFromCSV form utilizes Radio Button Functionality to allow the user to choose from either: CSV, HTML or Both. Finally, as this is the method in which the user would choose to process a CSV file, there is an OpenFileDialog component also integrated into the form. This allows the user to select a file from their computer when the *Select File* openfiledialog button is clicked.

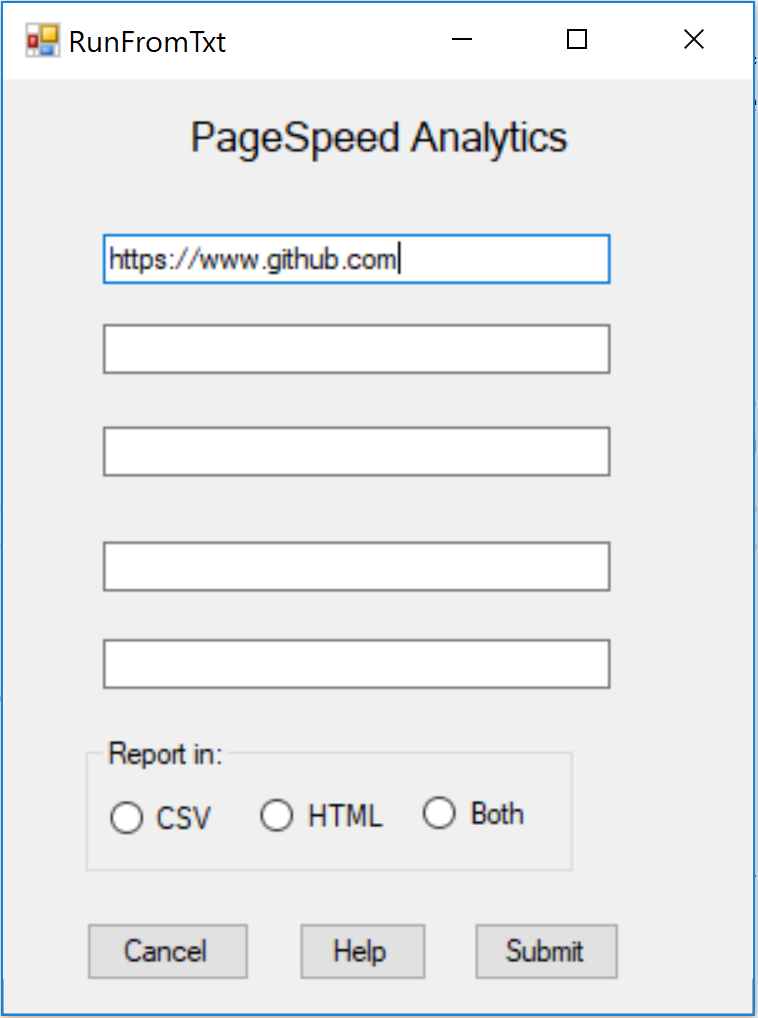
##### 

##### Figure 4: Run From CSV GUI Interface, using the RunFromCSV Windows Form

*See Section 2.3.9 for the class and method information for the RunFromCSV class.*

#### 2.4.3 Run From Text GUI (RunFromTxt)

When the user prompts the Run From Text Process, the RunFromTxt GUI is displayed. This is the second of two methods that the user can choose from to use the PSAA application. As with the Run From CSV GUI, the Run From Text GUI also includes the basic Cancel, Help and Submit buttons. Additionally it also, like with the Run From CSV GUI, utilizes the radio button functionality for the selection of the report method.



##### Figure 5: Run From Text GUI Interface, using the RunFromTxt Windows Form

*See Section 2.3.10 for the class and method information for the RunFroTxt class.*

#### 2.4.4 Runner Progress GUI (RunnerProgress)

The Runner Progress GUI, is a much simpler interface in comparison to the previous three user interfaces, from the user’s perspective. It is specified in this way, as if seeing it from the computer’s perspective this is very different. The GUI itself only includes a Progress Bar Component and Status Label, that allows the user to receive updates on the process of the procedures and services running. It is either instantiated from the Run From Text GUI or from the Run From CSV GUI. This allows for a more unified look.

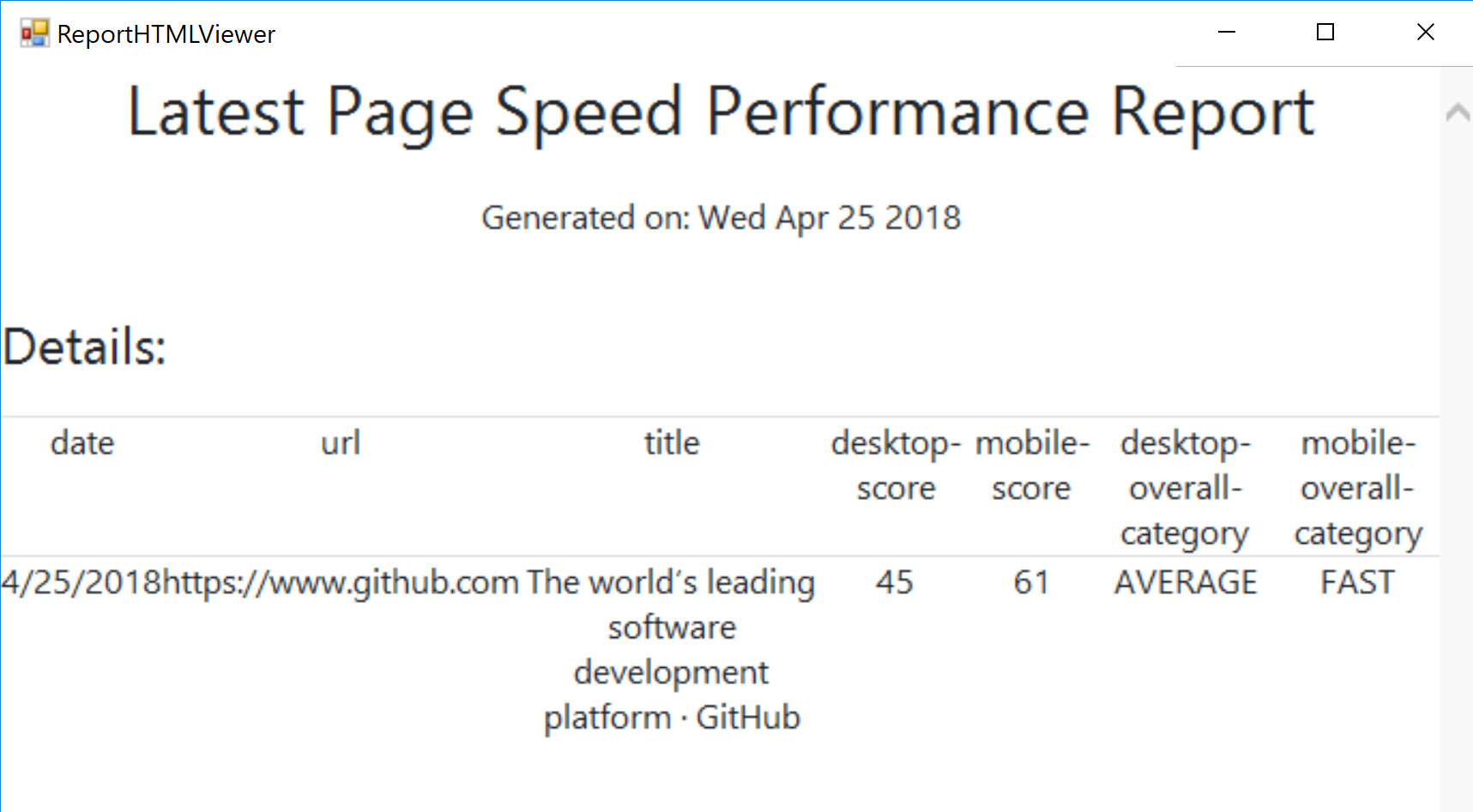
##### 

##### Figure 6: Runner Progress GUI Interface, using the RunnerProgress Windows Form

*See Section 2.3.11 for the class and method information for the RunnerProgress class.*

#### 2.4.5 HTML Report View GUI (ReportHTMLViewer)

The HTML Report View GUI is another simple class. It utilizes only one component, the Windows Form WebBrowser Component. This allows the HTML Report View GUI to create a web browser in the form to display the html report template file and write the new html text to it that contains the report information. Users can also print this report page, similar to that of a typical web browser. It is created, instantiated and run from the PageSpeedReport class.



##### Figure 7: HTML Report View GUI Interface, using the ReportHTMLViewer Windows Form

*See Section 2.3.2 for the class and method information for the ReportHTMLViewer class.*

# 

# References

Ashurst, D. (2012, August 19). Parsing a .CSV file in C#.NET. Retrieved from https://danashurst.com/parsing-a-csv-file/

Wood, J. (2012, July 4). Reading and Writing CSV Files in C#. Retrieved from https://www.codeproject.com/Articles/415732/Reading-and-Writing-CSV-Files-in-Csharp

Google (2018, January 18). Using page speed in mobile search ranking [Blog post]. Retrieved from <https://webmasters.googleblog.com/2018/01/using-page-speed-in-mobile-search.html?hl=en-US&utm_source=PSI&utm_medium=incoming-link&utm_campaign=PSI>.

Google. “PageSpeed Insights API Client Library for .NET | API Client Library for .NET | Google Developers.” Google PageSpeed Insights API Client Library for .NET, Google, https://developers.google.com/api-client-library/dotnet/apis/pagespeedonline/v4.