# **Design Section**

### 1 Outline System Design

### System Flow charts

**Current system flow chart**

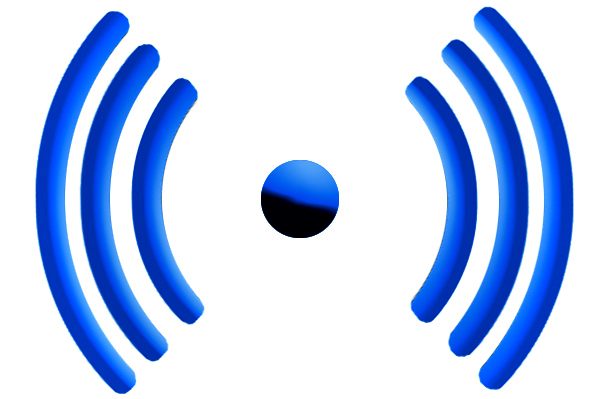
ISP Server

Router

Home computer

RJ45 Cable

Wi-Fi protocol



Tablet

Laptop

Smartphone

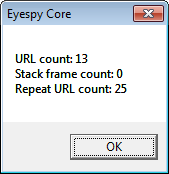
### User Interface Design

***The technicalities of the following user interfaces are given inside of the Systems Maintenance document. See section 2.5.***

**2.1 Eyespy Core UI**

C:\Users\Charles\Desktop\Capture.PNG

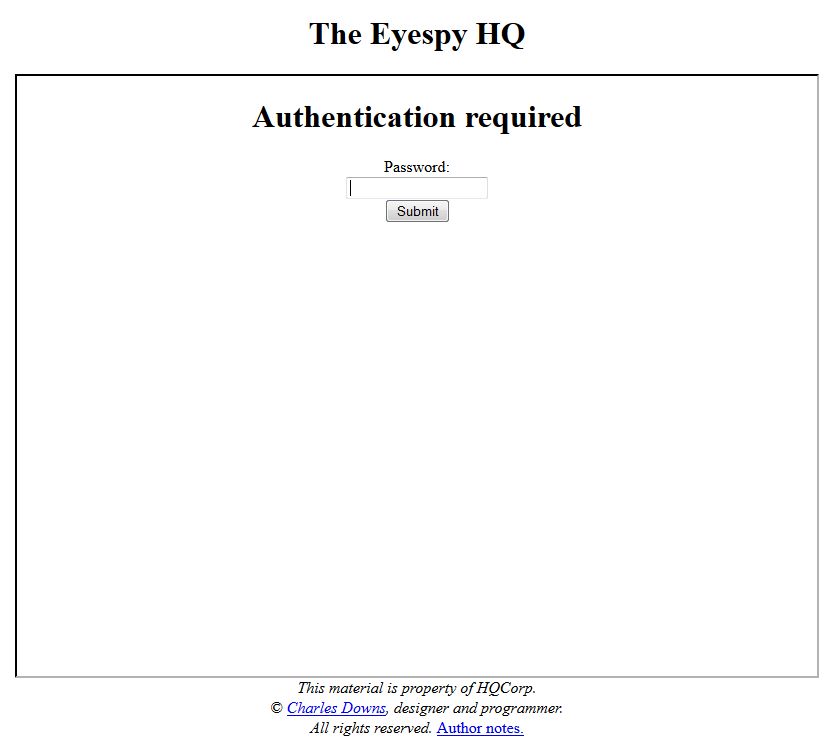
Current layout of the system tray icon.



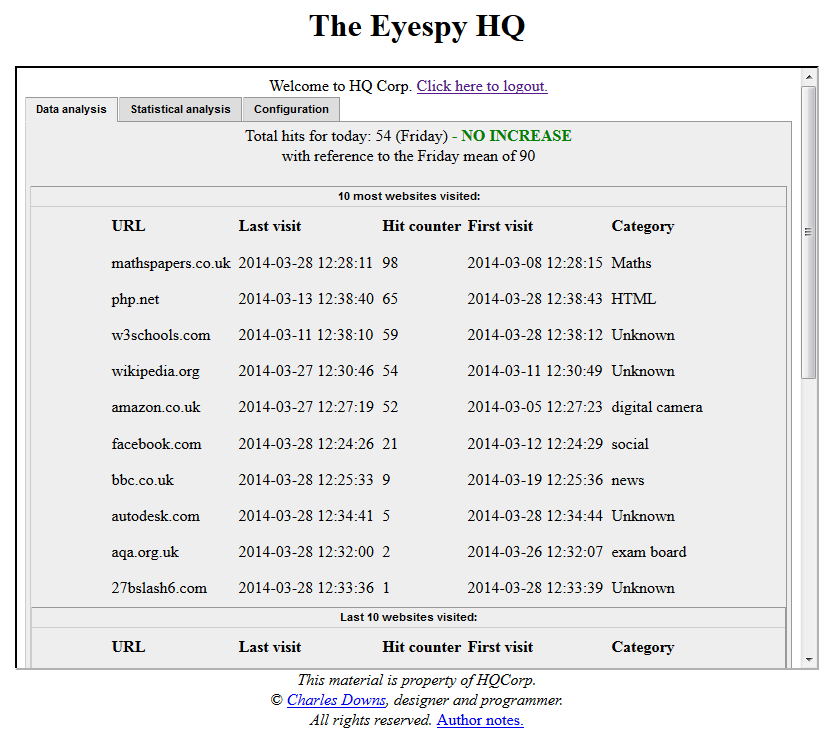
Above demonstrates the “on-click” event launched by the system tray icon. It details the current stack frame and repeated URL’s.

**2.2 Eyespy HQ UI**

**2.2.1 Authentication page.**



Password form and submit form button.

**2.2.2 Data analysis tab**

Tabbed content selectors

Collapsible content panel (**COLLAPSED**)

Collapsible content panel (**NOT** **COLLAPSED**)

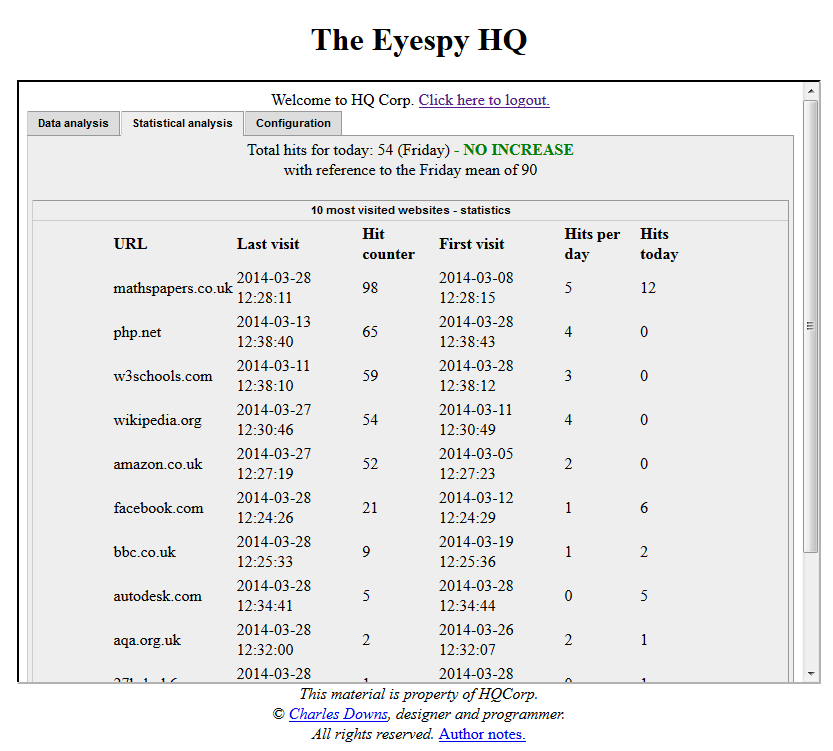
Data relative to which collapsible panel is collapsed

Logout link

Copyright notice

Configuration panel, used to edit the database

**2.2.3 Data analysis tab.**



Statistical data, relating to each website in the row. This is the only area in which differ the “Data analysis” tab from the “Statistical Analysis”.

**2.2.4 Configuration tab.**

 Delete all website entries from the database

Delete all category entries from the database

Reset all the hitcounters for each website

### 3 Hardware Specification

Hardware specification of my client:-

RAM: 4Gb

Processor: Intel(R) Core™ i5 @ 2.50GHz

Hard drive: 500GB SATA

### 4 Program structure – Core Hierarchy

Eyespy Core

Networking

HTTPReader

Strip Metadata from HTLM dump

Strip string from HTML dump

Destroy socket

Receive data

Create socket

HTTPWriter

Write to instance

Instantiate HTTPWRITER

Send data via HTTP string.

* Submit.php

***Full class definitions can be seen in SECTION 6.***

***Full API and annotated source code can be found in the Systems Maintenance.***

### 5 Design Data Dictionary

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Description | Data type  Access/Java | Size or range |
| Website | Website record in database holding fields such as Hitcounter, Category, etc.. | Database record | ~0.5Mb |
| Website | Website name | Text / String | Varchar(256) |
| IP | IP of website | Inet.addr (dots and numbers) | Depending on IPv4 or IPv6 either 2^128 or 2^64 |
| URL | URL of the website | Text / String | Varchar(256) |
| Category | Category under which the website is | Text / String | Varchar(64) |
| Hitcounter | Total amounts of hits the website had | Integer | 4 bytes |
| meanVisits | How many times the website has been visited over the set period of time | Integer | 4 bytes |
| lastVisit | Date at which the website was last visited | Datetime | 4 bytes |
| firstVisit | Date at which the website was first visited | Datetime | 4 bytes |
| Hits\_today | Total hitcount for the lastVisit date | Datetime | 4 bytes |

✓

### 6 Class definitions

**Eyespy HQ**

**CLASS** websiteEntries

**PRIVATE:**

Sql: SQL-instance object  
**PROTECTED:**

top\_10\_id: integer array

top\_10\_url: string array

top\_10\_category: string array

top\_10\_hitcounter: integer array

top\_10\_lastvisit: datetime array

top\_10\_firstvisit: datetime array

top\_10\_lastvisit\_id: integer array

top\_10\_lastvisit\_url: string array

top\_10\_lastvisit\_category: string array

top\_10\_lastvisit\_hitcounter: integer array

top\_10\_lastvisit\_lastvisit: date array

top\_10\_lastvisit\_firstvisit: date array

**PUBLIC:**

\_\_construct(parameter: sql): method (constructor)

get\_top\_10\_ids: function, returns integer array

get\_top\_10\_url: function, returns string array

get\_top\_10\_hitcounter: function, returns integer array

get\_top\_10\_category: function, returns string array

get\_top\_10\_firstvisit: function, returns datetime array

get\_top\_10\_lastvisit: function, returns datetime array

get\_10\_lastvisit\_url: function, returns string array

get\_10\_lastvisit\_hitcounter: function, returns integer array

get\_10\_lastvisit\_lastvisit: function, returns datetime array

get\_10\_lastvisit\_firstvisit: function, returns datetime array

get\_top\_10\_lastvisit\_category: function, returns string array

**PRIVATE** members of the websiteEntries class restrict access to member-functions of the websiteEntries class.

**PROTECTED** members of the websiteEntries class restrict access to member-functions of the websiteEntries class AND inheriting child classes.

**PUBLIC** members of the websiteEntries class are accessible by class instances of websiteEntries. They are used to return the value of **PROTECTED** members of the websiteEntries class.

This class is used to get fetch all website data from the database. It is purely an SQL wrapper class.

**CLASS** websiteStatistics **EXTENDS** *websiteEntries*

**PRIVATE:**

Sql: SQL-instance object

**PROTECTED:**

top\_10\_most\_visited\_mean: integer array

top\_10\_last\_visited\_mean: integer array

top\_10\_most\_visited\_today: string array

total\_hits\_today: integer

**PUBLIC:**

\_\_construct(parameter: sql): method (constructor)

get\_top\_10\_most\_visited\_mean(parameter: *websiteEntries*

instance): function, returns integer array

get\_top\_10\_most\_visited\_today(parameter: *websiteEntries* instance): function, returns string array

get\_10\_most\_visited\_today: (parameter: *websiteEntries* instance and *websiteStatistics* instance): function, returns string array  
get\_total\_hits\_today: function, returns integer

has\_usage\_increased\_today: method, prints boolean

**PRIVATE** members of the websiteStatistics class restrict access to member-functions of the websiteStatistics class.

**PROTECTED** members of the websiteStatistics class restrict access to member-functions of the websiteStatistics class AND inheriting child classes.

**PUBLIC** members of the websiteStatistics class are accessible by class instances of websiteStatistics. They are used to return the value of **PROTECTED** members of the websiteStatistics class.

This class used inheritance and containment to access data from its parent class: websiteEntries. It performs arithmetic and data comparison instructions to return statistics regarding protected members of the websiteEntries class.

**Eyespy Core**

**CLASS** Systray **EXTENDS** *Form*

**PRIVATE STATIC:**

sock: Socket

trayIcon: NotifyIcon (Microsoft imbedded-class)

trayMenu: ContextMenu (Microsoft imbedded-class)

urlcount: integer

repeatURLCount: integer

blockedURLCount: integer

**PRIVATE:**

bResponceBytes: function

**STATIC:**

bResponceBytes: byte array

stack: Stack (Eyespy class)

reader: HTTPReader (Eyespy class)

writer: HTTPWriter (Eyespy class)

Main: function

OnClientRecieve(parameter: IAsyncResult): function

**PUBLIC:**

Systray: function (class constructor)

**PROTECTED:**

OnLoad(parameter: EventArgs): function

About(parameter object, EventArgs): function

About(parameter bool): function

Systray\_Load(parameter sender, EventArgs): function

This is the main program class, within this class is the main program loop. It will utilize the classes listed below.

**CLASS** HTTPWriter

**PUBLIC:**

submitToHQ(string URL, string meta): function

HTTPWriter is composed of a single function used to submit data to the database once processed by the Systray class and collected by the HTTPReader class defined hereunder.

**CLASS** HTTPReader

**PUBLIC:**

getURL(parameter: string data): function, returns string

getMetadata(parameter string url): function, returns string

getBetween(parameter string start, string source, string end): function, returns string

Systray uses the Stack class to check if data has been processed within the current stack frame of size defined as a parameter in the constructor.

**CLASS** Stack

**PRIVATE:**

stackPointer: integer

stackSize: integer

stackItems: string array

**PUBLIC:**

stackFrameCount: integer

Stack(parameter int stackSize): constructor

isMemorized(parameter string stackItem): boolean

push(parameter: string stackItem) function

### 7 Data structures

Stack implementation to check for resubmitted domains that are not relevant to browsing because they could source ad servers or image servers, which are not consistent with the actual data. This allows reducing the number inconsistent entries by at least 75% because of repetition eliminated by checking stack.

Stack dump example when repeat URL is found:

[1] Google.com

[2] Facebook.com

[3]Twitter.com

[4]ad.google.com

[5]ad.google.com 🡨 Stack pointer

isMemorized(ad.google.com) evaluates to true and the URL is ignored.

***For a more technical stack dump, see the testing document.***

### 8 File organisation

In the program (Eyespy Core) a debug mode will be present; it will be able to be made active via a Boolean variable in the source code. If the debug mode is on, the program will write to 2 textfiles: log.txt and dump.txt.  
Log.txt will hold all the log data written by the program itself at runtime. This is useful to trace errors or program execution trace.  
The dump.txt will hold the entire HTML received on that day. Note that this file will become **very** huge very quickly, for this reason it will be deleted every 24 hours.

### 9 API used

HTMLAgilityPack – used to read various parts of data from an HTML input string, in the case of this project: meta tags.

This is a modification ported to the Core program as of March 2014, since the expected PHP function used to perform meta-getting procedures is not able to authenticate with modern websites due to a validity handshake between client and website.

### 10 Entity-relationship diagram

**Database tables:**

Website, Category,, DynamicStatistics, StaticStatistics.

**Relationships:**

Each Entries (can have) MULTIPLE Categories NORMALIZED via the WebsiteCategoryLink table.

**Table structure:**Entries (ID, hitCounter, lastVisit, URL)

Category(cID, category)

WebsiteCategoryLink(wID, cID)

**ER Diagram:**

Category

WebsiteCategoryLink

Entries

DynamicStatistics

StaticStatistics

### 11 Normalised database tables

The relationship between the Entries table and the Categories table is a many to many (Analysis 4.1, Design 10) – which means I’ve had to normalize the database via a link table: WebsiteCategoryLink.   
This table is structured in the following way-   
WebsiteCategoryLink(ID, cID, wID)

Where ID is the ID of the link and is primary key.

cID is the Category ID and is foreign key referencing the Category table.

wID is the Website ID and is foreign key referencing the Entries table.

I’ve designed this table to me in 3rd normal form, as:

1. There are no repeating groups or attributes (hence 1st normal form)
2. There are no partial key dependencies, (established via the link table) and is in 1NF (hence 2nd normal form)
3. The database meets the criteria for both 1NF and 2NF, and there are no non-key dependencies (hence in 3rd normal form).

For the SQL report, see Systems Maintenance section 2.

Table Name: entries

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Attribute Name | Primary key?  Foreign key? | Sample data | Data type  MYSQL | Data type | Size or range or other validation required |
| ID | PK, index | 1 | BIGINT | INT | 2^8 |
| URL | n/a | [www.google.com](http://www.google.com) | Varchar(256) | String | 256 max |
| hitcounter | n/a | 123 | INT | INT | 2^8 |
| Lastvisit | n/a | 2014-03-02 15:30:44 | DATETIME | String | n/a |
| firstvisit | n/a | 2014-03-02 15:30:44 | DATETIME | String | n/a |
| Hits\_today | n/a | 123 | INT | String | n/a |

Table Name: categories

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Attribute Name | Primary key?  Foreign key? | Sample data | Data type  MYSQL | Data type | Size or range or other validation required |
| ID | cID PK | 1 | INT | INT | 2^8 |
| category | n/a | “Gaming” | Varchar(256) | String | n/a |

Table Name: WebsiteCategoryLink

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Attribute Name | Primary key?  Foreign key? | Sample data | Data type  MYSQL | Data type | Size or range or other validation required |
| ID | PK, index | 1 | INT | INT | 2^8 |
| cID | Foreign key | 14 | INT | INT | 2^8 |
| wID | Foreign key | 123 | INT | INT | 2^8 |

Table Name: dynamicStatistics and staticStatistics

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Attribute Name | Primary key?  Foreign key? | Sample data | Data type  MYSQL | Data type | Size or range or other validation required |
| ID | PK, index | 1 | INT | INT | 2^8 |
| day | n/a | 1 *(Monday)* | INT | INT | 2^8 |
| Value | n/a | 123 | INT | INT | 2^8 |

Table Name: Logon

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Attribute Name | Primary key?  Foreign key? | Sample data | Data type  MYSQL | Data type | Size or range or other validation required |
| ID | PK, index | 1 | BIGINT | int | 2^8 |
| Username | n/a | [admin](http://www.google.com) | Varchar(256) | String | 256 max |
| Password | n/a | Admin | Varchar(256) | String | 256 max |

### 12 Algorithms

12.1 Algorithms pseudo code

**Main system algorithms:**

Algorithm to return the string between two other strings from a block.

**function** getBetween(string strSource, string strStart, string strEnd) returns string

integer Start, End

IF (strSource.Contains(strStart) && strSource.Contains(strEnd))

Start = strSource.IndexOf(strStart, 0) + strStart.Length

End = strSource.IndexOf(strEnd, Start)

return strSource.Substring(Start, End - Start)

ENDif

Else return ""

Regex validation to ensure the data being handled is in alphanumeric format.

**function** isClean(data) returns string

IF(data matches regular expression:'/^[a-z0-9 .\-]+$/i')

return true

ENDif

else return false

Main data handling algorithm, executed whenever data is received by the socket.   
This algorithm will ensure the validity of inbound data.

**procedure** onClientReceive(AsyncState)

IF(bytes\_received count >= 40)

data = convert to UTF8(bytes\_recieved)

IF(data start with “GET”)

URL = get URL from data

IF(URL != NULL and URL length <= 64)

IF(URL is not in current stack frame)

Keywords = get keywords from URL

Submit URL and Keywords to database handler

ENDif  
ENDif ELSE invalid data

ENDif

else return false

A function of the HTTPReader class – makes use of the getBetween algorithm to get the location of the URL, nested between both parameters declared hereunder.   
**function** getURL(data) returns string

Return getBetween(data, "Host: ", "User-Agent:")

A function of the HTTPReader class – uses the HtmlAgilityPack API to extract the keywords from a website using a valid handshake, as required by modern websites (making the get\_meta\_tags() PHP function obsolete, as discussed in the Systems Maintenance, section 2.6).

**function** getMetadata(URL) returns string array

webGet = new HtmlWebGet  
document = webGet.load(URL)

metaTags = document.selectNodes(“//meta")

IF(metaTags != NULL)

Foreach(tag in metaTags)

IF(tag.Name == “keywords”)

Keywords = tag

ENDif

ENDForeach

Keywords = Split into array using “,” as delimiter.

Return Keywords[1], Keywords[2], Keywords[3]

ENDif, ELSE return No Meta Tags

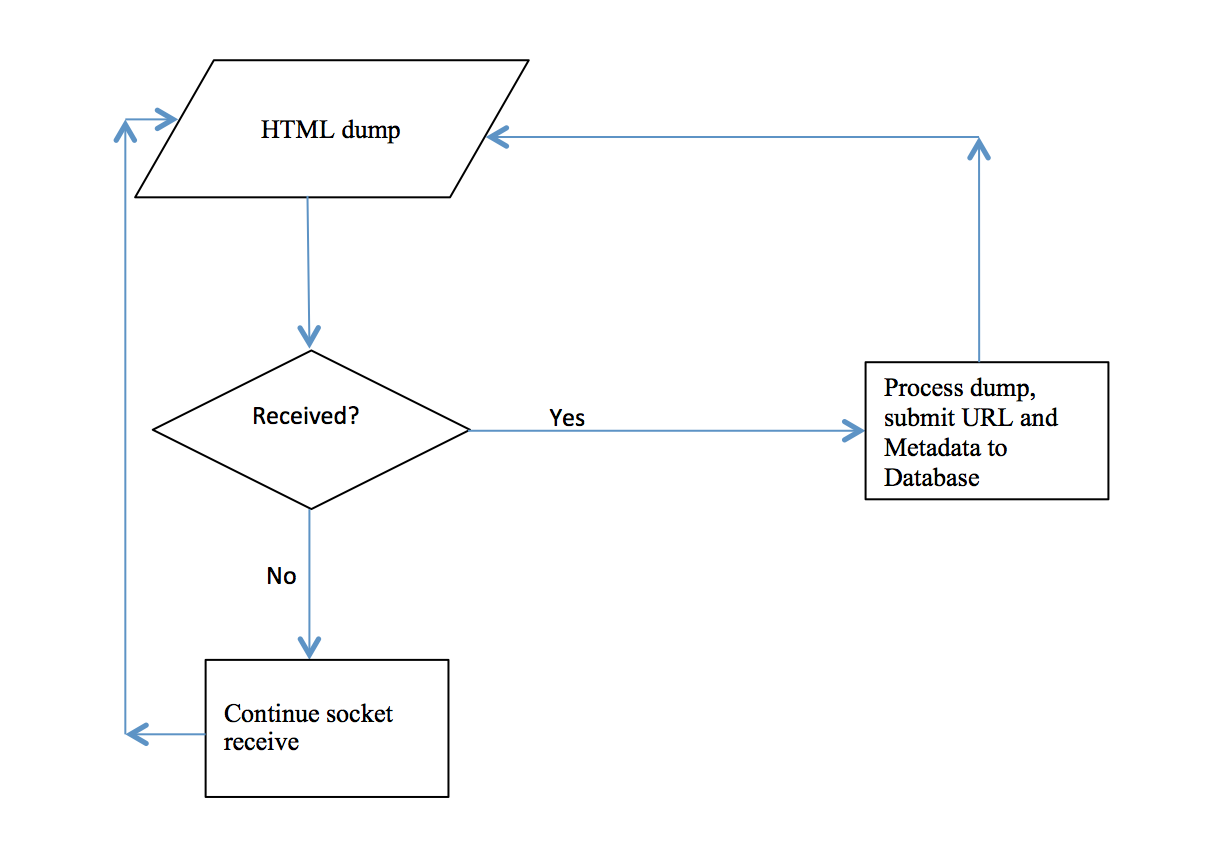
A function of the HTTPWriter class – submits URL and get keywords, sending them to the Submit.php script.

**procedure** submitToHQ(URL, keywords)

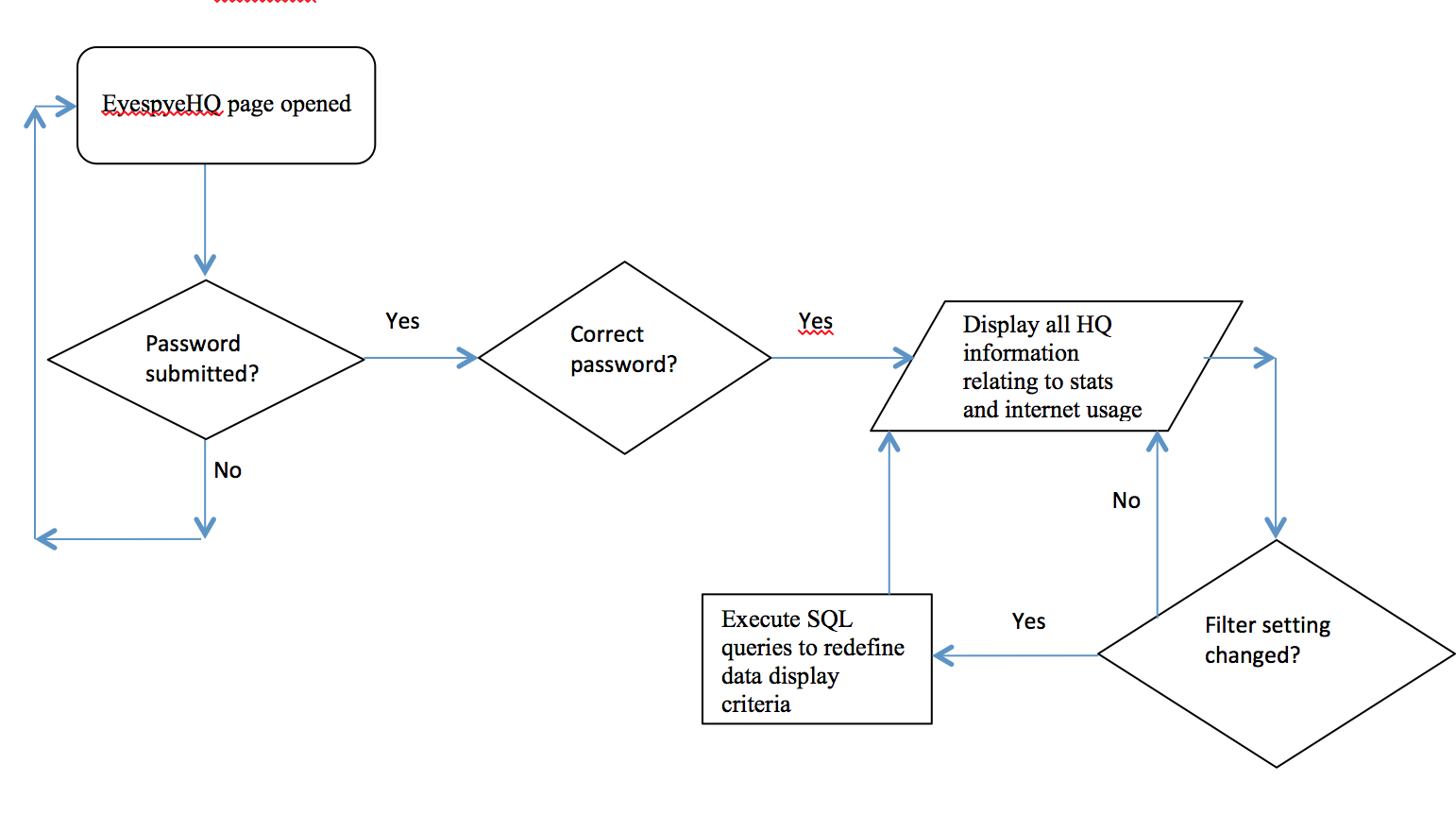
WebRequest = new WebRequest(<http://www.charlesnet.info/eyespy/submit.php>)  
WebRequest.Method = POST  
data = “URL=URL&keywors=keywords”  
using StreamWriter { ( write data to output stream )  
WebRequest.Send(stream data) }

12.2 Flow charts

Eyespy Core



Eyespy HQ



### 13 Queries

**Inserting website record when first visit (variables prefixed with $):**

INSERT INTO entries (URL, lastvisit, hitcounter) VALUES ('$URL', '$today', '1')

**Getting top 10 visited websites:**

SELECT \* FROM entries ORDER BY hitcounter DESC LIMIT 10

**Getting 10 last visited websites:**

SELECT \* FROM entries ORDER BY lastvisit DESC LIMIT 10

**Find if database holds records for a specific URL ($url):**

SELECT \* FROM entries WHERE URL = '$URL' LIMIT 1

**Updating website record when page visited (variables prefixed with $):**

UPDATE entries SET hitcounter = '$hit' WHERE ID = '$id'

UPDATE entries SET lastvisit = '$today' WHERE ID = '$id'

UPDATE entries SET hits\_today = '$hits\_today' WHERE ID = '$id'

**Getting dynamic statistics for any day of the week (where days are 1 to 7):**

SELECT value FROM staticStatistics WHERE day = ‘$day’ limit 1

### 14 Detailed design of printed output

In the case of this project it will not be necessary, not by standards nor by user request, to have any program-printed reports, all data (statistical and analytical) will be output to the screen in a standardized table form, thus, allowing the user to copy any required data (not that this expected to be done) to an excel document.

### 15 Preliminary test plan

Testing will involve

1. boundary data – expected to pass (or not) validation for size
2. valid data – expected to pass validation (or not) for type
3. malicious data – data that could harm the database, that will be blocked
4. database test – all data that enters the program should be correctly inserted into the database with correct associated values entered or updates (date, hitcounter, url, visits).

### 16 Detailed test data

To test data, a data-dumping program has been designed. Its purpose is to handle **high** volumes of data to the processing application that will then store data to the database.

The advantage of such a program is that it will allow testing large volumes of data at a very high speed, data can be small, large and sent at any rate. An example would be: small data sent at frequent intervals, large data sent at small intervals, and any such combination. This also will allow testing performances of the data processor (Eyespy HQ) and evaluating any improvements that may need to be made, should the response time or performance benchmarks be unsatisfactory. For live testing, see the HTTPWriter folder in the root project folder.

This program will also allow to test the robustness of the HQ module before going into in-depth design of the Core module, so this “simulation” will be able to tell me if there are any major faults in the preconception of the Core module, rather than be faced with major errors amidst a large package.

The current spec for the HTTPWriter is as follows:

* Can have URL/text input
  + URL has to match regex pattern, this field is mandatory
  + Text input will be for keywords, this field is not mandatory
* Output will be in the form of short text error messages
  + Regex validation failed: “regex\_no”
  + Entry updated for existing website “entry\_updated”
  + Entry crated for new website “entry created for %s”
* Output will be generated by the Submit.php file pertaining to the HQ module.
* Input will be sent to HQ for validation and submission.