# **Design Section**

### 1 Outline System Design

### 1.1 System Flow charts

### **Eyespy Core**

### 

Continue socket receive

Received?

Process dump, submit URL and Metadata to Database

HTML dump

Yes

No

### **Eyespy HQ**

EyespyeHQ page opened

Correct password?

Display all HQ information relating to stats and internet usage

Password submitted?

Yes Yes

No

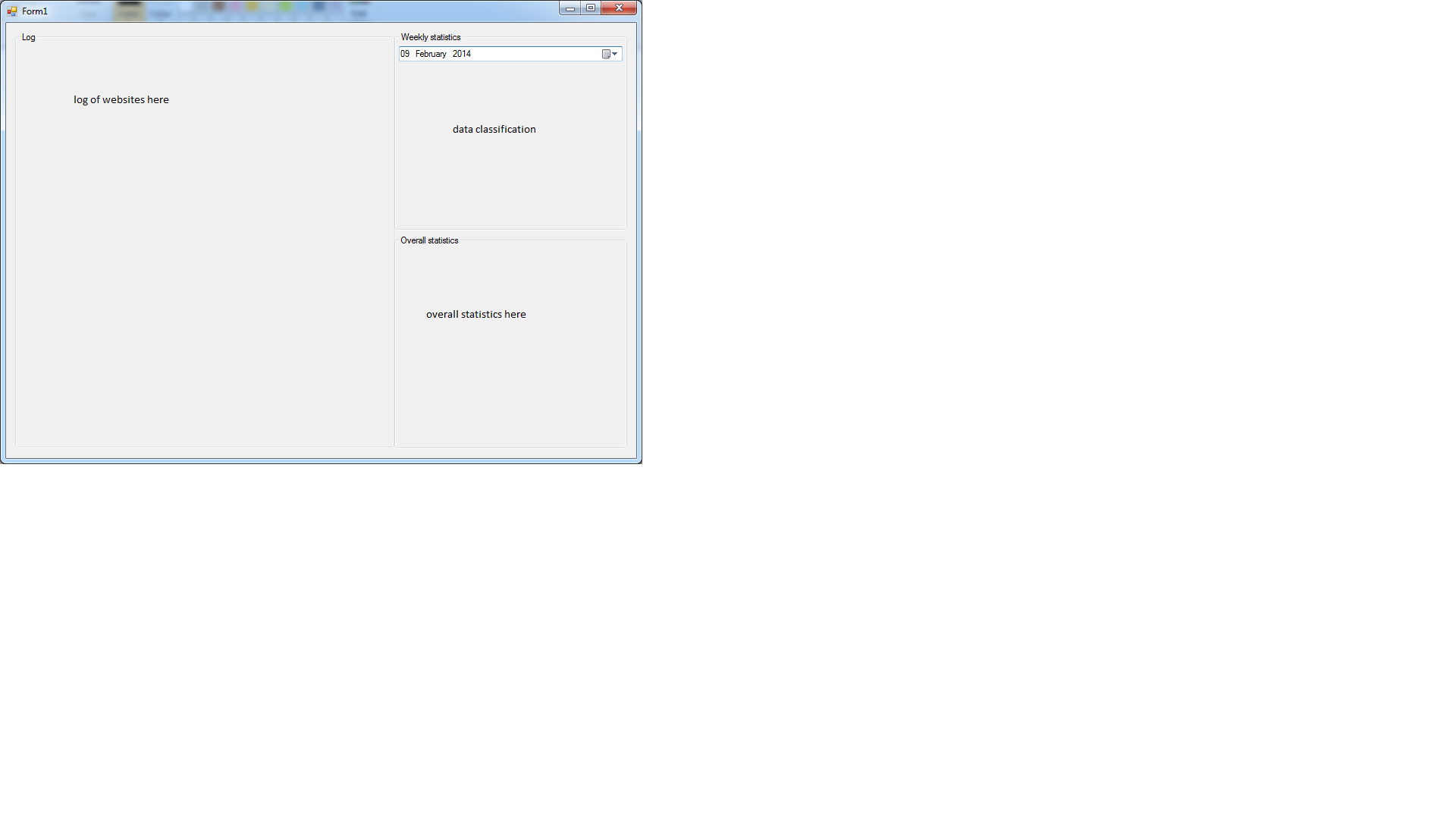
No

Filter setting changed?

Execute SQL queries to redefine data display criteria

Yes

### 2 User Interface Design



Log of websites = output

Data classification = output from input form above

Overall statistics = output

### 3 Hardware Specification

Hardware specification of my client:-

RAM: 4Gb

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

Hard drive: 500GB SATA

### 4 Program structure

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

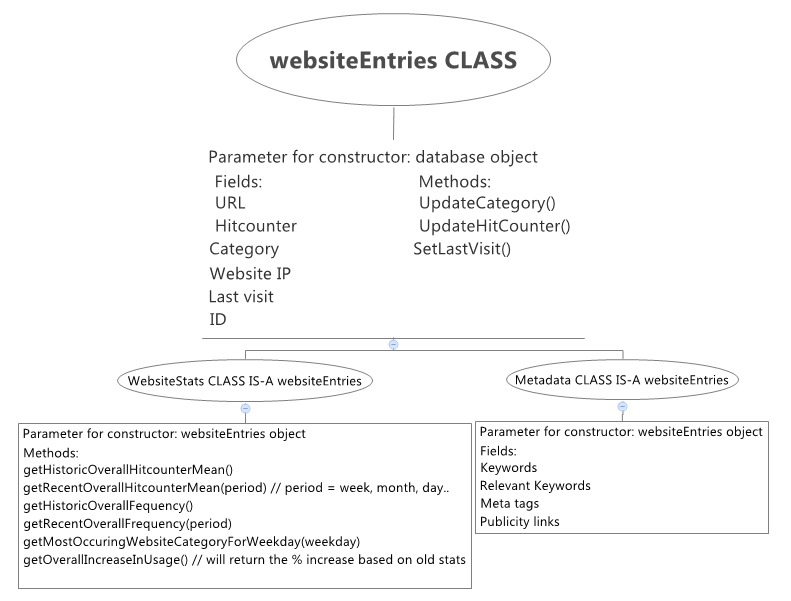
Send data via HTTP string.

* Update hitcounter
* Update metadata
* Update stats
* Inset website record
* Inset metadata
* Insert statistics
* …

### 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 |

### 6 Object diagrams and class definitions



The websiteEntries class holds all the information relating to one website entry in the database. An instance of this class will be created when the user requests to view a website summary.

### 7 Data structures

Stack implementation to check for resubmitted subdomains that are not consistent with the actual data. This allows reducing the number inconsistent entries by at least 75%.

### 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

N/A

### 10 Entity-relationship diagram

**Database tables:**

Website, Category, Metadata, DynamicStatistics, StaticStatistics.

**Relationships:**

Each Entries (can have) ONLY ONE Metadata NORMALIZED via the WebsiteCategoryLink table.

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

Each Entries (can have) ONLY ONE Metadata.

Each Metadata (can have) MULTIPLE Websites.

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

Category(cID, category)

Metadata(mdID, keywords, category, metaTags)

WebsiteCategoryLink(wID, cID)

**ER Diagram:**

WebsiteCategoryLink

Category

Entries

Metadata

DynamicStatistics

StaticStatistics

### 11 Normalised database tables

Complete this section if you will be connecting to a database. Discuss how you set about normalising your tables and complete a table setting out the attributes for each table/relation. Get your normalised database design checked and approved – this can be a stumbling block for some students so make sure you do this in good time before deadlines. Make sure data types are appropriate to the type of database and to Java (or Android).

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 | FK | [www.google.com](http://www.google.com) | Varchar(256) | String | 256 max |
| hitcounter | n/a | 123 | BIGINT | Int | 2^8 |
| Lastvisit | n/a | 2014-03-02 15:30:44 | DATETIME | String | n/a |

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

C#

public static string getBetween(string strSource, string strStart, string strEnd)

{

int 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);

}

else

{

return "";

}

}

PHP   
function isClean($data){

if(preg\_match('/^[a-z0-9 .\-]+$/i', $data)) {

return true;

} else return false;

}

function getRequest($data) {

return substr($data, 17);

}

### 13 Queries

INSERT INTO website (URL, IP, lastVisit) VAUES (var, var, var)

UPDATE website SET lastVisit = var WHERE website.ID = ID

UPDATE website SET hitCounter = website.hitCunter+1 WHERE website.ID=ID

### 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 will be done) to an excel document.

### 15 Preliminary test plan

See p257 for an example of an outline and detailed test plans.

### 16 Detailed test data

To test data, a data-dumping simulator has been conceived. Its purpose is to hand **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 to test 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 to test performance of the data processor (Eyespy HQ) and evaluate 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.

