Coursework for:

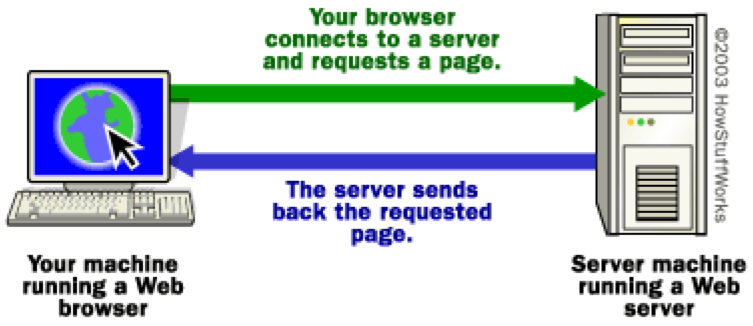
Internet Application Programming
3ISE517

MalN
2007xxx

CONTENTS

Introduction

Every time we request a web page on our browser, the browser would have created a connection with a web server, sent a request and received the required page before displaying it on our screens. A basic representation of such a request in shown in the diagram below:


Your browser connects to a server and requests a page.
The server sends back the requested page.
Your machine running a Web browser
Server machine running a Web server
©2003 HowStuffWorks

[Reference: http://computer.howstuffworks.com/web-server1.htm, 23/11/08]

Generally pages requested by web browsers are static web pages which would be held in the web server's local repository. All the web server has to do is, locate the requested web page and return it to the browser. What one means by a static web page is that its content does not change until its owner modifies it.

But some applications or scenarios require information not from pages that have been pre-written but from pages that are generated based on user input. For example, when searching for an article on the web, the results are generated based on the search words entered by the user. Such web pages are known to be generated dynamically. Being able to produce output based on what the user has produced, makes web servers much more interactive. In these situations, the server would actually have to process the information and generate a page to send back to the user based on request. Web servers achieve this by the use of the Common Gateway Interface.

Before moving onto how dynamic pages are created, the following section explains how a static web page would be requested and retrieved from a server.

Basic HTTP Requests

Every time a user requests a webpage by clicking on a link or simply typing it into the address bar, the web browser would break down the URL into three main components:

- The protocol , eg: http, ftp, https
- The server, eg: www.google.com
- And the filename, eg: /file.html

Every web server has an IP address and a domain name. Using the IP address, the web browser would connect to the server. After the connection is formed, following the HTTP protocol, the browser sends a GET request, asking to retrieve the webpage. The server would then process the request, locate the required page and return the result. The server sends back a HTTP response to the browser, containing the HTML code for the webpage. The web browser would then interpret the HTML tags and display the web page.

Using the Common Gateway Interface

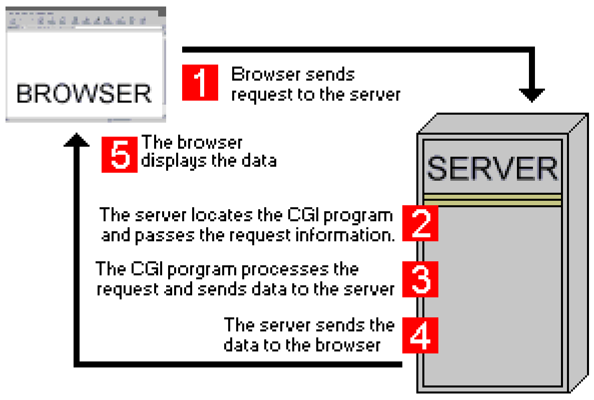
The Common Gateway Interface (CGI) is a protocol which defines the rules for transmitting information between a web server and an external program. According to webopedia.com, any program that is designed to accept and return data that conforms to the CGI specification can be classified as a CGI program. Using such programs, allow web servers to dynamically interact with the users. CGI scripts can be written in any programming languages, such as C, Perl, Java, etc.

Basic Overview of the use of CGI:

Using the database example mentioned on the web page at hoohoo.nesa, how CGI is used will be explained:

If a person wants to share his database with the world by connecting to the web, he will require a CGI program to transmit information between the web browser and the database engine. When a client requires some information from the database, his web browser sends request to the server. The database engine using CGI executes a CGI script which would then process the request and return the results to the web browser. As the script is being processed on the web server itself, this technique is known as server-side scripting.

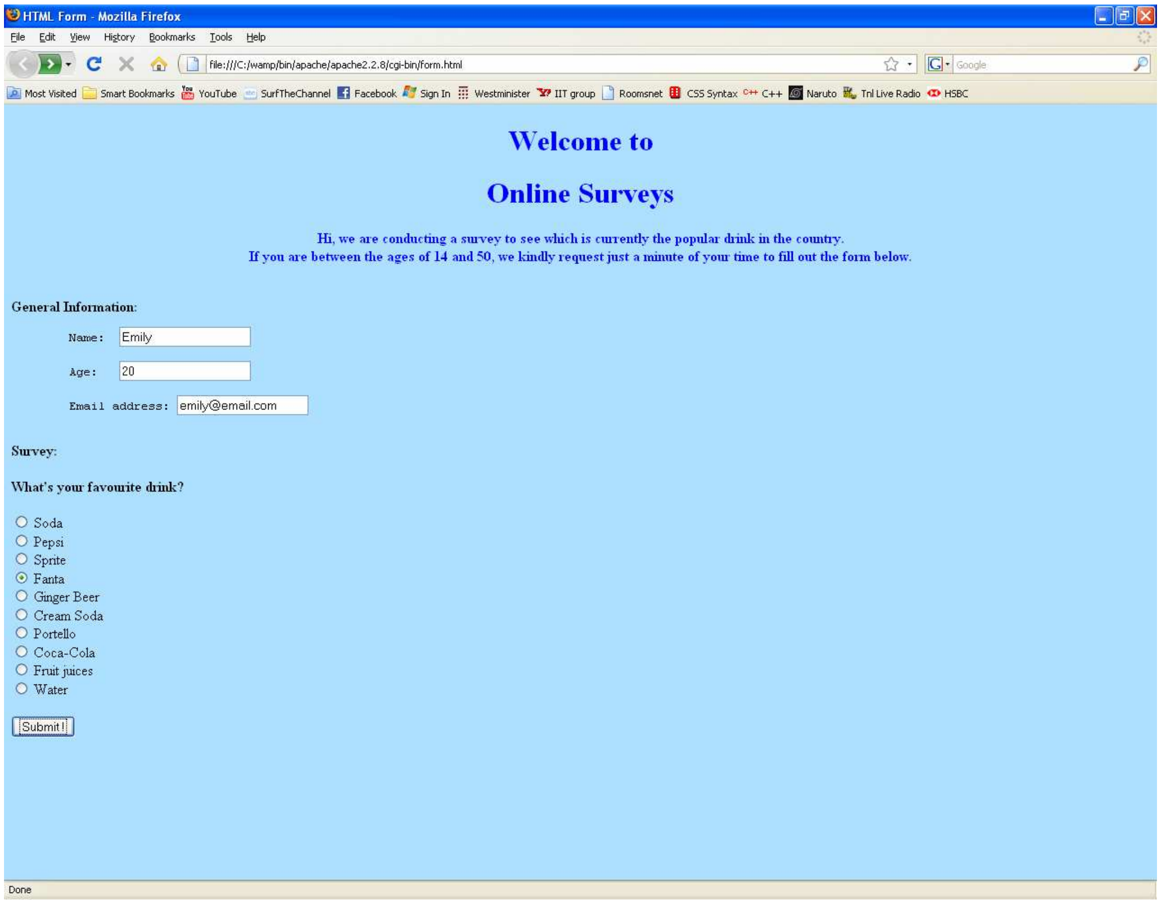
The diagram below illustrates the use of CGI:


BROWSER
1 Browser sends request to the server
5 The browser displays the data
SERVER
The server locates the CGI program and passes the request information. 2
The CGI porgram processes the request and sends data to the server 3
The server sends the data to the browser 4

[Reference: http://www.webdevelopersnotes.com/basics/client_server_architecture.php3, 23/11/08]

Explanation of how CGI is used:

Generally the input for a CGI script is received from a HTML form on a webpage. For example, the form shown below:



Welcome to

Online Surveys

Hi, we are conducting a survey to see which is currently the popular drink in the country.
If you are between the ages of 14 and 50, we kindly request just a minute of your time to fill out the form below.

General Information:

Name: Emily

Age: 20

Email address: emily@email.com

Survey:

What's your favourite drink?

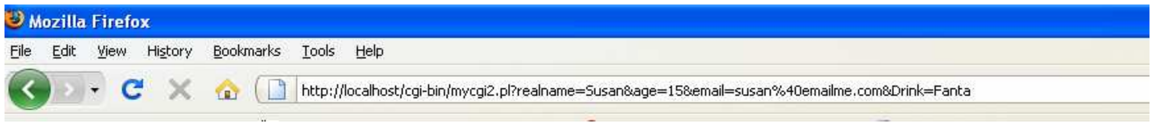
Soda
Pepsi
Sprite
[✓ Fanta]
Ginger Beer
Cream Soda
Portello
Coca-Cola
Fruit juices
Water

Submit!

The webpage simulates that of a basic online survey page. The form is created to take in the 4 separate items of data. Once this form is filled and the user presses the Submit button at the bottom of the page, a HTTP request would be sent to the server. Upon submission, the file specified in the action attribute of the form would be requested from the server. This form would request a CGI script on submission.

The information that the user has entered in the form can be submitted by 2 methods: GET and POST. By sending the form data using the GET method, the data is appended to the URL of the CGI program and therefore all the data is visible in the address bar in name/ value pairs.

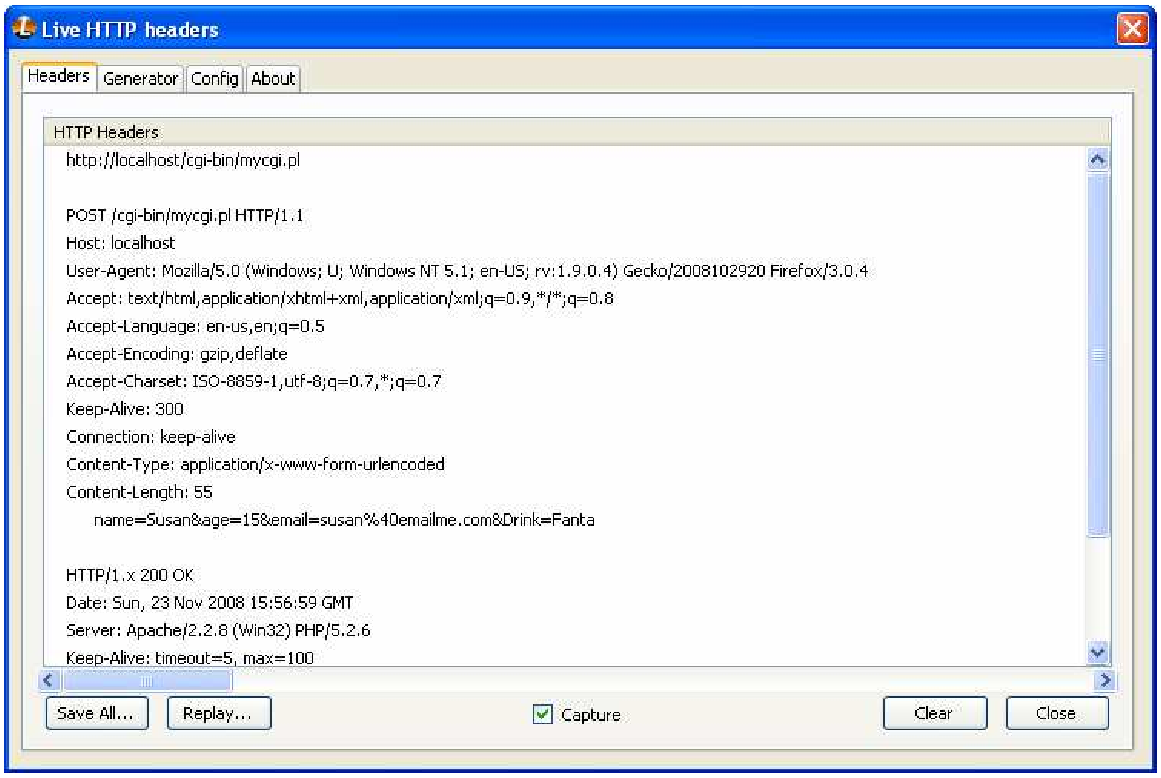
Shown below is how the address bar would look if the GET method was used:



```
http://localhost/cgi-bin/mycgi2.pl?realname=Susan&age=15&email=susan%40emailme.com&Drink=Fanta
```

In the POST method, the data is added to the HTTP request message. If the data sent needs to be secure, for example, the username and password for an email account, the POST method is the suitable method. For this form, I have used the POST method to send the data.

Shown below is the http request that is sent on submitting the form:

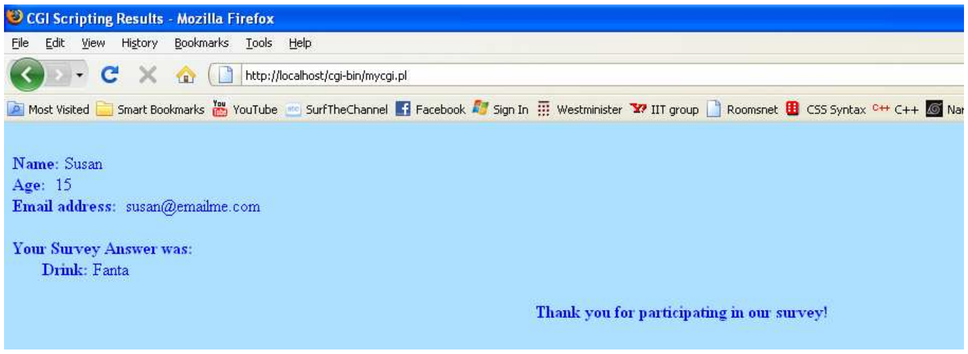

Live HTTP headers
Headers Generator Config About
HTTP Headers
Save All... Replay... Capture Clear Close

```
http://localhost/cgi-bin/mycgi.pl

POST /cgi-bin/mycgi.pl HTTP/1.1
Host: localhost
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.9.0.4) Gecko/2008102920 Firefox/3.0.4
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 300
Connection: keep-alive
Content-Type: application/x-www-form-urlencoded
Content-Length: 55
    name=Susan&age=15&email=susan%40emailme.com&Drink=Fanta

HTTP/1.x 200 OK
Date: Sun, 23 Nov 2008 15:56:59 GMT
Server: Apache/2.2.8 (Win32) PHP/5.2.6
Keep-Alive: timeout=5, max=100
```

Upon receiving the request from the client's browser, the server executes the CGI Script, which is located in the cgi-bin directory. The script then begins to process the request. As all the data is packaged up when being sent to the script, it has to be separated into their respective parts. Once each data has been split, the script can begin various other processes, such as validation, etc. After its processing, the resulting output can be returned back to the web browser in the form of HTML code.

In the form I have created, upon submission the mycgi.pl file is requested from the server. This script uses the split function to separate the various data items. Each data is validated. A web page is then dynamically generated from the script, displaying either the error messages if any field had been incorrectly filled in, or the data that the user has entered into the survey.

The webpage that is returned to the user upon completing the survey correctly is shown below:


CGI Scripting Results - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://localhost/cgi-bin/mycgi.pl
Most Visited Smart Bookmarks YouTube SurfTheChannel Facebook Sign In Westminister IIT group Roomsnet CSS Syntax C++

Name: Susan
Age: 15
Email address: susan@emailme.com

Your Survey Answer was:
Drink: Fanta

Thank you for participating in our survey!

REFERENCED SITES:

How Web Servers Work - The Basic Process

http://computer.howstuffworks.com/web-server1.htm

How Web Servers Work - Extras: Dynamic Pages

http://computer.howstuffworks.com/web-server12.htm

How Web Servers Work – Behind the Scenes

http://computer.howstuffworks.com/web-server2.htm

How CGI Scripting Works - The CGI Mechanism

http://computer.howstuffworks.com/cgi2.htm

How CGI Scripting Works - Forms: Sending Input

http://computer.howstuffworks.com/cgi4.htm

How CGI Scripting Works - Summary

http://computer.howstuffworks.com/cgi7.htm

Common Gateway Interface

http://hoohoo.ncsa.uiuc.edu/cgi/intro.html

The Client-Server Architecture

http://www.webdevelopersnotes.com/basics/client_server_architecture.php3

Web and Database Integrating Using CGI

http://www.ncsi.iisc.ernet.in/raja/netlis/wise/cgi/webcgi.htm

CGI

http://www.webopedia.com/TERM/C/CGI.html

How CGI Scripting Works - The CGI Mechanism

http://computer.howstuffworks.com/cgi2.htm

Web Server

http://www.webopedia.com/TERM/W/Web_server.html

APPENDIX

APPENDIX A:

Program listings of Form.html

```
<!--
Written by : Malshani Nanayakkara
StudentID: 2007020
-->

<html>
<head><title>HTML Form</title></head>

<body bgcolor="#ADDFFF">

        <font color="blue">
        <h1><center>Welcome to</center></h1>
        <h1><center>Online Surveys</center></h1>

        <p align=center><b>Hi, we are conducting a survey to see which is currently the popular
drink in the country.<br/>
        If you are between the ages of 14 and 50, we kindly request just a minute of your time to
fill out the form below.</b></p></font>

        <br />

        <form method=POST ACTION="http://localhost/cgi-bin/mycgi.pl">

                <b>General Information:</b> <br />
                <pre>
        Name:  <input type=text name="name"><br />
        Age:   <input type=text name="age"><br />
        Email address: <input type=text name="email">
                </pre>

                <b>Survey: <br /><br />
                What's your favourite drink? </b>
                <br /><br />
                <input type=radio name="Drink" value="Soda">  Soda </input><br />
                <input type=radio name="Drink" value="Pepsi">  Pepsi </input><br />
                <input type=radio name="Drink" value="Sprite">  Sprite </input><br />
                <input type=radio name="Drink" value="Fanta">  Fanta </input><br />
                <input type=radio name="Drink" value="Ginger Beer">  Ginger Beer
</input><br />
                <input type=radio name="Drink" value="Cream Soda">  Cream Soda
</input><br />
                <input type=radio name="Drink" value="Portello">  Portello </input><br />
                <input type=radio name="Drink" value="Coca-Cola">  Coca-Cola </input><br
/>
                <input type=radio name="Drink" value="Fruit Juices">  Fruit juices </input><br
/>
```

```
                <input type=radio name="Drink" value="Water">  Water </input><br />
                <br />
                <input type=submit value="Submit !">

        </form>

</body>
</html>
```

Program listing of mycgi.pl

```
#!C:/Perl/bin/perl.exe
use CGI qw(:standard);

read(STDIN, $data, $ENV{CONTENT_LENGTH});
print "Content-type: text/html\n\n";

#unpack the form data
($firstpair, $secondpair, $thirdpair, $fourthpair) = split(/&/, $data);
($firstkey, $name) = split(/=/, $firstpair);
($secondkey, $age) = split(/=/, $secondpair);
($thirdkey, $email) = split(/=/, $thirdpair);
#($thirdkey, $phoneno) = split(/=/, $thirdpair);
($fourthkey, $drink) = split(/=/, $fourthpair);

$validentry = 0;  #keeps track if all entries are valid

print "<html><head><title>CGI Scripting Results</title></head>";
print "<body bgcolor=\"#ADDFFF\"><font color=\"blue\"><br/>";

#-------------------------VALIDATION----------------------------------

#validating name
        if($name !~ /[A-Za-z]+/ || $name =~/[0-9A-Za-z]+[0-9]+/ )                    #if it is outside
the range of the alphabet or it is letters mixed with number
        {
                print "<br/>ERROR: Name given contains an invalid character or has not been
provided!";
                $validentry =1;
        }
        else
        {
                $name1 = $name;           #keeps an original version so that it can be written file. It
can read easily from the file then.
                $name1 =~s/\+/ /;
                print "<b>Name:</b> $name1";
        }

#validating age
        if ($age<14 || $age>=50)
        {       print "<br />ERROR: Age is not between 14-80!";
                $validentry =1;  }
        else
        {       print "<br/><b>Age:</b>&nbsp; $age"; }

#validating email address
        if (!($email =~/^[\w]+%40[\w]+/))
        {       print "<br/>ERROR: Incorrect email address ";
                $validentry =1;  }
        else {
                $email =~ s/%40/@/;             #replaces the %40 with an @ sign
                print "<br/><b>Email address:</b>&nbsp; $email"; }

#validating survey answer
        if ($drink!~/[\w]/ )
        {       print "<br/>ERROR: You haven't selected an answer!";
                $validentry =1;}
        else
        {
                $drink1 = $drink;
                $drink1 =~s/\+/ /;
                print "<br/><br/><b>Your Survey Answer was:</b> ";
                print "<br/><b>&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp; Drink:</b>
$drink1";  }

#displays a message if all entries have been correctly submitted
if ($validentry==0)
{
        print "<br/><br/><center><b>Thank you for participating in our survey!</b></center>";

                #writing results to file
        $file = 'survey_results.txt';
        open(OUTFILE, ">>$file");
                print OUTFILE "$name\t$age\t$email\t$drink\n";
        close OUTFILE;
}

print "</font><br/>";
print "<form method=POST action=\"http://localhost/cgi-bin/survey.pl\">";
print "<input type=\"submit\" value=\"View survey results\" ";
print "</form>";
print "</body></html>";

#-------------extra code----------
#validating telephone number
#       if ((length ($phoneno)==10 ) && ($phoneno =~ /^[0-9]+$/) )
#       { print "\nTelephone No: $phoneno"; }
#       else
#       {       print "\nERROR: invalid phoneNo!"; }
```

APPENDIX B:

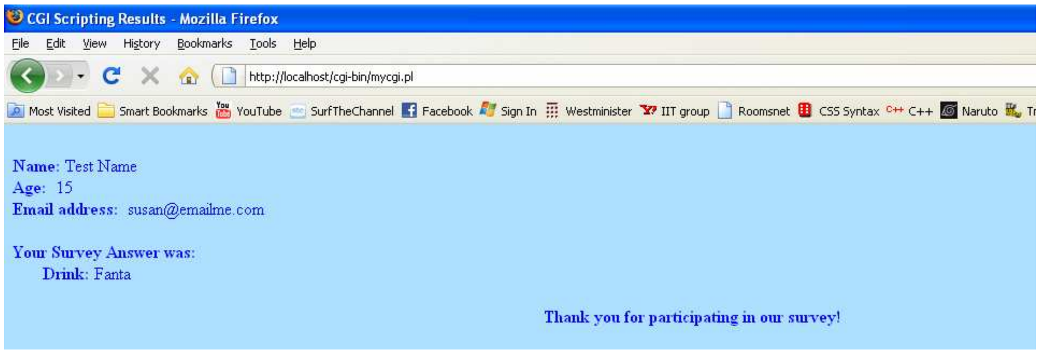
Screenshots of validating user input:

The following tests can be performed:

1. Two names written
2. No name provided
3. Number written instead of the name
4. Letters and numbers mixed
5. No age provided
6. Age given outside the required range
7. Letters written instead of a number
8. No email address given
9. Email address without the '@' sign
10. No option selected

But as the same error message come for each test, the screenshots for certain validations are shown below.

Test 1: Two names written

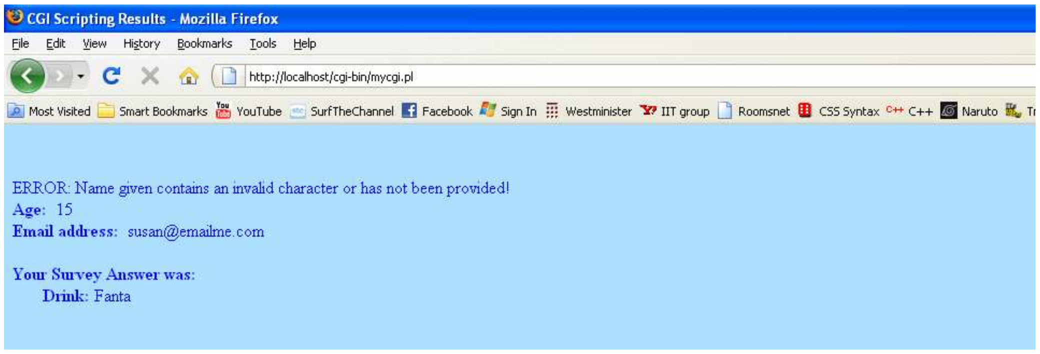

CGI Scripting Results - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://localhost/cgi-bin/mycgi.pl
Most Visited Smart Bookmarks YouTube SurfTheChannel Facebook Sign In Westminister IIT group Roomsnet CSS Syntax C++ Naruto

Name: Test Name
Age: 15
Email address: susan@emailme.com

Your Survey Answer was:
Drink: Fanta

Thank you for participating in our survey!

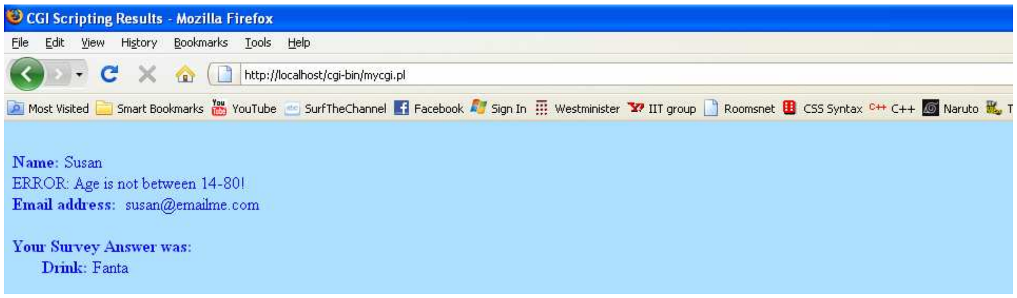
Test 2 - 4: Invalid Input for the Name


CGI Scripting Results - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://localhost/cgi-bin/mycgi.pl
Most Visited Smart Bookmarks YouTube SurfTheChannel Facebook Sign In Westminister IIT group Roomsnet CSS Syntax C++ Naruto

ERROR: Name given contains an invalid character or has not been provided!
Age: 15
Email address: susan@emailme.com

Your Survey Answer was:
Drink: Fanta

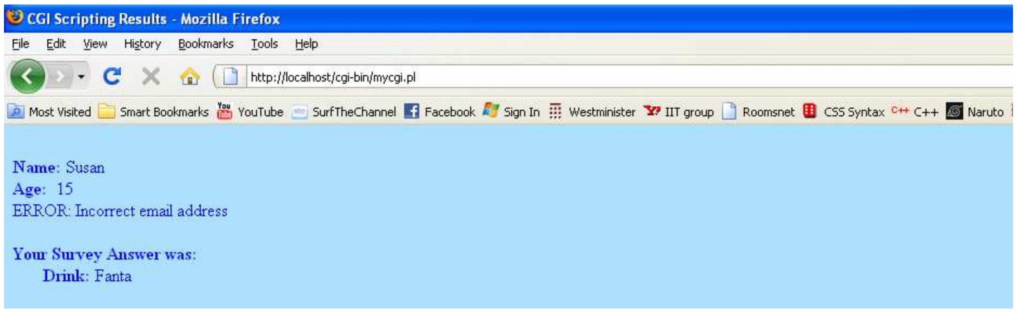
Test 5 - 7: Invalid Input for the Age


CGI Scripting Results - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://localhost/cgi-bin/mycgi.pl
Most Visited Smart Bookmarks YouTube SurfTheChannel Facebook Sign In Westminister IIT group Roomsnet CSS Syntax C++ Naruto

Name: Susan
ERROR: Age is not between 14-80!
Email address: susan@emailme.com

Your Survey Answer was:
Drink: Fanta

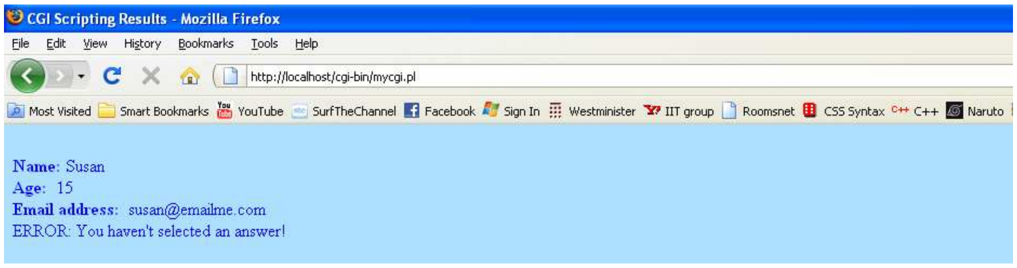
Test 8-9: Invalid Input for the Email address


CGI Scripting Results - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://localhost/cgi-bin/mycgi.pl
Most Visited Smart Bookmarks YouTube SurfTheChannel Facebook Sign In Westminister IIT group Roomsnet CSS Syntax C++ Naruto

Name: Susan
Age: 15
ERROR: Incorrect email address

Your Survey Answer was:
Drink: Fanta

Test 10: No Selection for the Survey Question


CGI Scripting Results - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://localhost/cgi-bin/mycgi.pl
Most Visited Smart Bookmarks YouTube SurfTheChannel Facebook Sign In Westminister IIT group Roomsnet CSS Syntax C++ Naruto

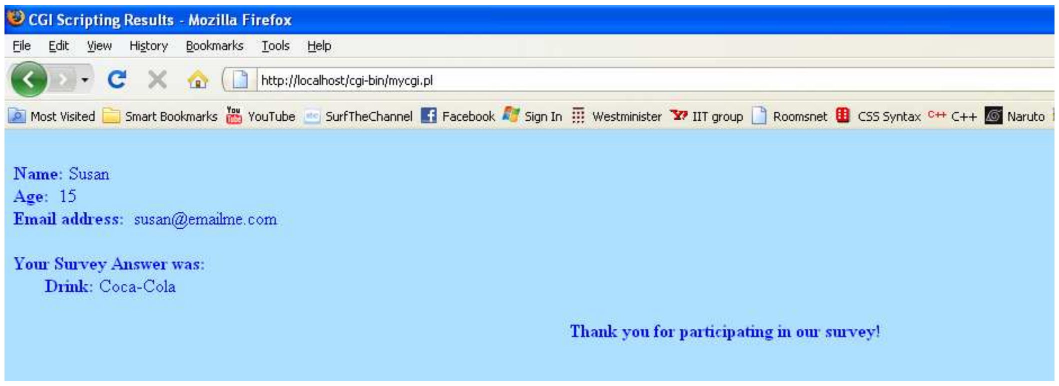
Name: Susan
Age: 15
Email address: susan@emailme.com
ERROR: You haven't selected an answer!

APPENDIX C:

Additional Features:

-writes to a file

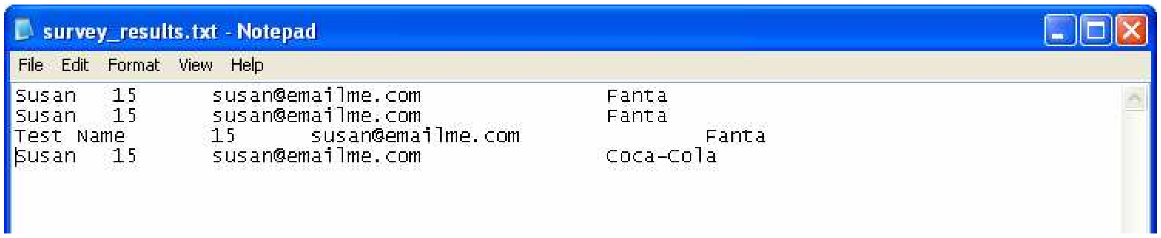
If the user has correctly entered data into each of the 4 fields, the information is written on to a text file called 'survey_results.txt'.


CGI Scripting Results - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://localhost/cgi-bin/mycgi.pl
Most Visited Smart Bookmarks YouTube SurfTheChannel Facebook Sign In Westminister IIT group Roomsnet CSS Syntax C++ Naruto

Name: Susan
Age: 15
Email address: susan@emailme.com

Your Survey Answer was:
Drink: Coca-Cola

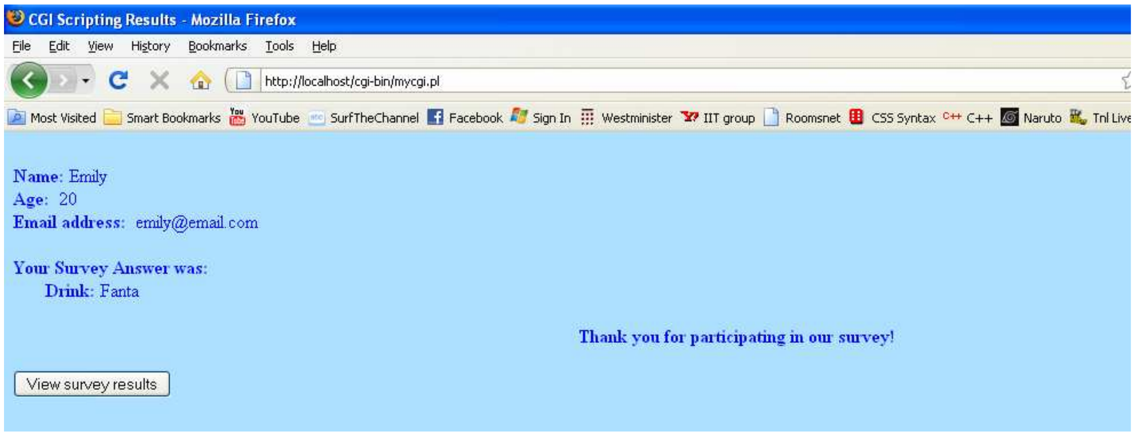
Thank you for participating in our survey!



```
Susan    15        susan@emailme.com                 Fanta
Susan    15        susan@emailme.com                 Fanta
Test Name          15        susan@emailme.com                Fanta
Susan    15        susan@emailme.com                 Coca-Cola
```

- reads from the text file and displays survey results

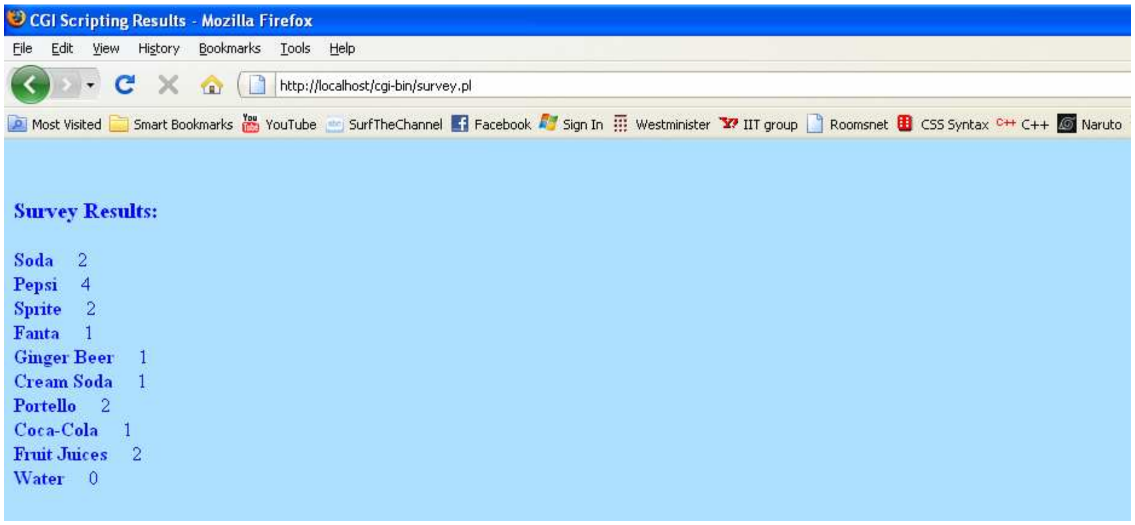
Using this text file, the survey results can be viewed:


CGI Scripting Results - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://localhost/cgi-bin/mycgi.pl
Most Visited Smart Bookmarks YouTube SurfTheChannel Facebook Sign In Westminister IIT group Roomsnet CSS Syntax C++ Naruto Tnl Live

```
Name: Emily
Age: 20
Email address: emily@email.com

Your Survey Answer was:
     Drink: Fanta

                                        Thank you for participating in our survey!
```

The page that mycgi,pl returns, contains a button which runs another script which will read the file and display the survey results.

And a page like the one below will be generated:


CGI Scripting Results - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://localhost/cgi-bin/survey.pl
Most Visited Smart Bookmarks YouTube SurfTheChannel Facebook Sign In Westminister IIT group Roomsnet CSS Syntax C++ Naruto

```
Survey Results:

Soda   2
Pepsi   4
Sprite   2
Fanta   1
Ginger Beer   1
Cream Soda   1
Portello   2
Coca-Cola   1
Fruit Juices   2
Water   0
```

A program listing for the survey.pl file is given below:

Program listing of survey.pl

```
#!C:/Perl/bin/perl.exe
use CGI qw(:standard);

print "Content-type: text/html\n\n";
print "<html><head><title>CGI Scripting Results</title></head>";
print "<body bgcolor=\"#ADDFFF\"><font color=\"blue\"><br/>";

open(INFILE, 'survey_results.txt');
@drinks=("Soda", "Pepsi", "Sprite", "Fanta", "Ginger Beer", "Cream Soda", "Portello", "Coca-
Cola", "Fruit Juices", "Water");
@drinks_score=(0, 0, 0, 0, 0, 0, 0, 0, 0, 0);

while ($line=<INFILE>)          #reads file line by line
{
        ($name, $age, $email, $survey) = split(/\t/, $line);          #seperates each line into
seperate data

        $survey =~s/\+/ /;          #if the drink has two words, the '+' sign will be replaced by a
space

        $i=0;
        while ($i<10)
        {
                if($survey=~/$drinks[$i]/)
                {       $drinks_score[$i]++;    }                    #the line is compared with an
array containing all the drinks

                $i++;
        }

}

print "<br/><font size=\"+1\"><b> Survey Results:</b></font><br/><br/>";
$i=0;
while ($i<10)
{
        print "<b> $drinks[$i]</b>&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;$drinks_score[$i]";
        print "<br/>";
        $i++;
}

print "</body></html>";
```