**Djoumessi Benjamin Gautier**

**DESIGN**

The software is a web service that makes it possible for users to convert currencies of the European Central Bank(ECB) to and from Euro, on responsive web pages.

The author also makes it possible for other developers to access his data by use of an API with the following URL; [https://maviance/api /v1/ccAPI.php?format=X](https://maviance/api%20/v1/ccAPI.php?format=X), where ‘X’ represents the format in which the data will be displayed.

**Data.php**

This file basically retrieves data from the ECB website using API (application program interface) and stores the selected information into a MySQL database. This information is the name, short form and exchange rate against Euro. The selected information from data retrieved are then assigned to variables which are subsequently stored in a database table using a MySQL query. This file is simple and thus reduces possibility of errors.

**Easy\_transfer.php**

This is a responsive web page which contains a form, from where the user enters the amount he/she wants to convert and the currency to/from which he/she is converting. The given information is then sent to the currency\_converter.php file. The author made this page responsive by the use of the HTML “META” tag setting it as follows

“<META name="viewport" content="width=device-width, initial-scale=1, userscalable=yes"> “. “Width=device-width” makes the page to match the screen of the device on which it is being viewed, “initial-scale=1” creates a 1:1 relationship between CSS pixels and device independent pixels no matter the orientation of the device. It also allows the page to use the full landscape width.” userscalable=yes” allows the user to zoom on the page.

**ccAPI.php**

This is the file that actually retrieves the data previously stored in the MySQL database table. It first checks if is the “currency” and “amount” variables have been posted. If the parameters are passed, they are assigned to variables for proper use. If the parameter has not been passed, nothing happens and it throws an error message. If the parameter is passed, they are assigned to variables for proper use. A connection is then made to the MySQL database table, a MySQL query made to the table and the result gotten. The result is then formatted for display. There are two possible formats for the output, JSON and XML.

**Currency\_converter.php**

This is the web page on which the final information is rendered to the user. The code in this file contains both PHP an HTML. It receives the information of the user from the form in the “easy\_transfer.php” web page and makes use of the ccAPI.php described above to get the data from the database. It first checks if this parameter is received from the post form. If not, nothing happens and no information will be displayed. If it is, the posted information is assigned to variables. The data obtained from the URL, in JSON or XML, is then decoded accordingly and the appropriate information assigned to variables for display.

This page is made responsive in a similar way as the “select page.htm”, by the use of the HTML “META” tag and CSS in exactly the same way as above (select.htm).

**Db.php**

This is a simple php file which is require to make the connection to the database. It contains the login credentials to access the database used to design the web service.

**Client.css**

This is a simple CSS (cascading style sheets) file making the images displayed on the web pages responsive. This file makes the images adapt to the size of the screen of the device on which the web pages are being viewed.

**DIAGRAMATICAL REPRESENTATION OF SOFTWARE DESIGN.**

ECB XML file

MYSQL DATABASE

Data.php

Retrieves information from provider sites by use of API’s

Saves information in database.

ccAPI.php

Receives destination from the user and the format in which the data has to be encoded.

Makes connection to the MySQL database table and retrieves appropriate data.

Encodes data into appropriate format

Currency\_converter.php

Responsive web page. Gets information from easy\_transfer.php.

Uses ccAPI.php to retrieve information from MySQL database table.

Decode the data and display it with appropriate image.

Easy\_transfer.php

Responsive web page form to get information from the user

USER DEVICE BROWSER

**TESTING AND PERFORMANCE**

The software was tested by the author and the result stored in the table below;

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Test case | Description | Test data | Expected result | Actual result | Pass/Fail-Priority |
| DA001 | Check if required data is retrieved from provider sites and stored in database | data.php | Required data retrieved and stored | Required data retrieved and stored | Pass |
| DS001 | Check if destination is selected and sent to appropriate file | Destination (select page.htm) | Destination selected and sent | Destination selected and sent | Pass |
| RT001 | Check responsiveness of text content of web page | select page.htm, client.php | Text content responsive | Text content responsive | Pass |
| RP001 | Check responsiveness of pictorial content of page | Images on web pages (client.css) | Pictorial content responsive | pictorial content responsive | Pass |
| DR001 | Check if destination is received and passed to API’s URL | Destination  (client.php) | Destination received and passed to API’s URL | Destination received and passed to API’s URL | Pass |
| DD001 | Check if data is retrieved from database and displayed in appropriate format | API.php | Data retrieved and displayed in appropriate format | Data retrieved and displayed in appropriate format | Pass |
| ID001 | Check if appropriate information is displayed to the user | client.php | Appropriate information displayed to user | Appropriate information displayed to user | Pass |

Following the test results, the software attains all its objectives and thus has very high performance in adequate situation. This is according to the author’s objectives and requirements.

**EVALUATION**

The software attains its objectives quite alright in the perfect situation. Usability of the software is really easy and debugging is made simple in case any crash occurs as the files are well separated and made simple to understand with proper comments alongside the code. The web pages displayed to the user are well responsive providing a good user interface and experience regardless of the device used to access the pages. The author made the contents of the web pages are concise enhancing user experience. Also there is the possibility of external access to the software from other web services by use of an API. This API can provide data in both JSON and XML format thus providing greater availability and diversity. Equally the retrieved data from provider sites are stored in a database thus information is made available faster to the user. The API developed by the author is also easy to use with only two parameters needed to retrieve the data.

However, the software fails to give alternative actions in case one of its components fail and thus gives a poor handling of exceptions. Also the author has not provided any alternative in case the provider sites fail are not available at any time. The software also has a limited audience due to its restriction to journeys from Wolverhampton to a certain number of destinations. Also the developer’s server is subject to saturation due to the absence of developer keys for access to its data through his API, making him unable to monitor access to his data by other web services which can then make his web pages very slow, or even, to fail.

**AVAILABILITY OF DOCUMENTATION ON PROVIDER SITES**

The API’s used by the author where gotten for free from the website of their respective developers. The API for weather information was gotten from Open Weather Map. The URL for the API is; <http://api.openweathermap.org/data/2.5/weather?q=town&key=developerapikey>, where ‘town’ represents the place you want to get information about and ‘developerapikey’ is a password-like string of characters attributed to each user of the API by its developers to monitor access to their web service by for example restricting the number of elements retrieved per hour by a particular user. Usage of town name is an option among three and seemed to be the most appropriate for the author. There is also the possibility to get data in other format like XML or HTML but by default, the data is produced in JSON format.

The information about the journey to the final user’s destination from Wolverhampton was obtained using the Google Distance Matrix API which has the following URL ; <https://maps.googleapis.com/maps/api/distancematrix/format?origins=Wolverhampton&destinations=town&language=english&key=developerapikey>, where ‘format’ stands for the way you want your data to be encoded, that is either JSON or XML, ‘town’ stands for the user’s destination and ‘developerapikey’ has the same use as the on the URL for the open weather map API. The parameter ‘language’ is an optional parameter which makes it possible to choose the language in which you want you data to be displayed. This could then produce the necessary information needed by the author.

**REFERENCES**

1. Google Developers, the Google Distance Matrix API. Accessed [15th November, 2014]. Available at <<https://developers.google.com/maps/documentation/distancematrix>>.
2. Google Developers, web Fundamentals. Accessed [12th January, 2015]. Available at <<https://developers.google.com/web/fundamentals/layouts/rwd-fundamentals/set-the-viewport>>.
3. Open Weather Map, current weather data. Accessed [14th November, 2014]. Available at <<http://openweathermap.org/current>>.
4. Brenelz Web Solutions, how to create a simple API with PHP and MySQL. Accessed [10th January, 2015]. Available at <<http://www.brenelz.com/blog/how-to-create-a-simple-api-with-php-and-mysql>>.