Coursework 1 - Ciphers Report

Zsolt Varga
40212393@live.napier.ac.uk
Edinburgh Napier University - Web Technologies (SET08101)

Abstract

The Cipher Madness is a simple and responsive website implementing three different ciphers. As an assignment for Edinburgh Napier University, this project is trying to implement basic UX and Usability principles, and demonstrate clean and well structured code. The website does not use any libraries whatsoever.
The purpose of this paper is to document the development process of the project and to critically reflect on it. This report also contains and evaluates personal performance and challenges faced during development.

Keywords – Web, Technologies, Design, HTML, JS, JavaScript, CSS, Zsolt, Varga, Edinburgh, Napier, University, Ciphers, Web-page, Coursework, Front-end, Static

1 Introduction

The Cipher Madness website was created as an assignment for Edinburgh Napier University's Web Technologies module. The full assignment was to create a web application, containing at least two working ciphers - these should be able to encipher, as well as decipher messages. The website must also include a design document, reflecting all design choices made during development, and an index.html file that serves as a home page.

1.1 Cipher Choices

The choice of ciphers came down to coming up with ones that are sufficient enough in complexity, but not out of the scope of the module. Three ciphers were chosen in the end:

Transposition cipher - There are many interpretations of what a transposition cipher is, but all of them are based on placing the text in a grid, then reading the grid in a different way (eg. top-to-bottom vs left-to-right).

Sanctus Notis - The name of this cipher translates to "Holy cipher" and is closely related to the Ave Maria cipher invented by Trithemius. [1] This is a steganographic cipher, meaning the enciphered message acts as an ordinary message - in this case a Latin prayer.

Binary code - Considering binary code a cipher is a bit controversial, however, it satisfies all requirements of a cipher's behaviour. In fact, for someone who does not recognize ASCII, a mid-step in encoding text to binary, this cipher would prove to be difficult to decode and therefore would be highly efficient in a pre-computing era.

1.2 Design

When creating the website, the main objective was not only to simulate functional ciphers, but also to make the cipher pages aesthetically pleasing and straightforward to use. Each cipher page has a distinct theme and colour. The whole website is done in flat design, a navigation bar is present on every page, and the layout is kept consistent throughout every page. It is therefore safe to say there has been just as much effort put into the design as the cipher development, if not more.

2 Software Design

Before getting to any software implementation, it was important to sketch out the layout of the pages in order to identify similarities and better understand how we want the code to function. In this case, all cipher pages are given the same layout, therefore it is only sensible to expect some inheritance to occur. See Figure 1.

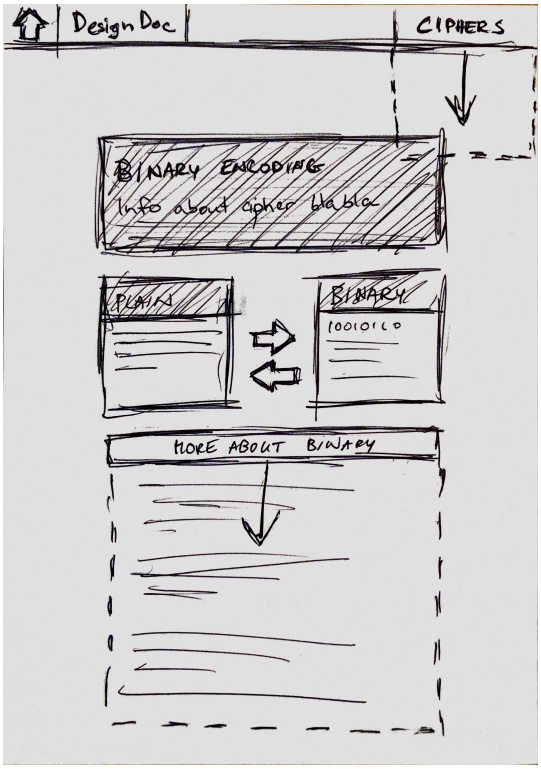

Figure 1: Shared layout of cipher pages

Also, in Figure 2, the index page shows the same navigation bar as the cipher pages, meaning that the navbar should be a stand-alone entity that is embedded in other pages.

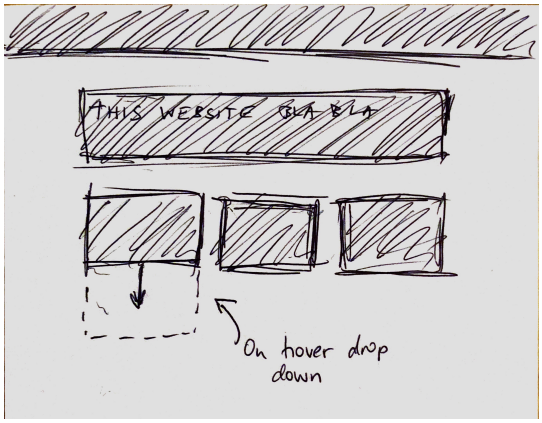

Figure 2: Layout of the home page

Having some experience in front-end web development as well as software development, it was clear that organisation within code as well as files was going to be very important. The project directory therefore contains several folders in which the pages are organised.

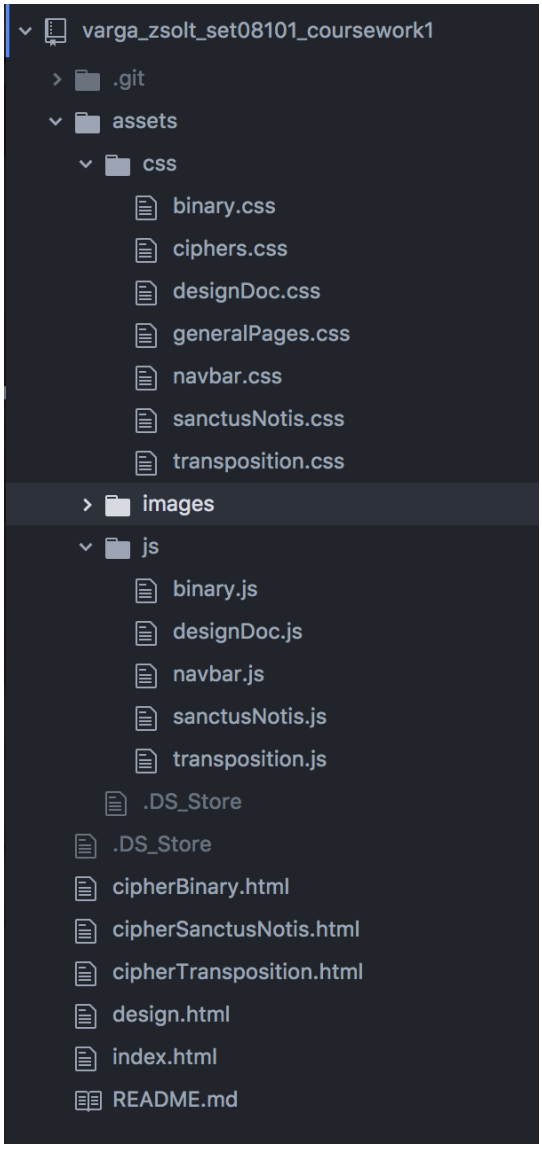

Figure 3: Organisation within repository

All .html files reside within the root directory (Figure 3). In the assets folder, each .html then has a corresponding .js and .css file placed in the js folder and css folder respectively. On top of this, there are some .css and .js files that are used by several/all pages. For instance, the layouts of all the cipher pages are nearly identical, therefore the code that formats this layout was placed in a ciphers.css file that all pages link to. There is no reason, however, for a cipher's colour formatting to be accessed by another cipher. Hence the binary.css, transposition.css etc. These contain styles exclusive for the cipher linked to them.

Looking deeper into the actual code, specifically the CSS formatting, there are certain similarities to the repository organisation that are important. First of all, since the ciphers.css file is used for "high level" formatting (implemented by every .html file), it mainly uses selectors with low specificity (class, tag-name). This is done so that the styles specified in this document act as a default setting but can be overridden very easily.
On the other hand, CSS files like binary.css have the very opposite purpose. They build on top of the more general stylesheets, but have the final word in the styling. This is due to the fact they use high specificity selectors - mainly id.

When considering which approach to take with the JavaScript files, it wasn't clear what would be the best option. Surely, all cipher pages have similar methods when it comes to encoding and decoding ciphers, therefore it would make perfect sense to create an abstract class for a cipher. In this case, in a project of this scale and low complexity, it seems more like an unnecessary complication rather than a solution. As a result, every cipher has its own standalone implementation with no dependencies. Similarly, the navbar has its own JavaScript document containing all the functionality.

3 Implementation

The most time-consuming part of the actual implementation was the CSS. There were many considerations that had to be kept in mind, since every .html links to several style sheets (See Figure 4). Within the CSS files, it was important to keep the specificity of final styles higher, so there is a rather great amount of style overrides going on in the background.

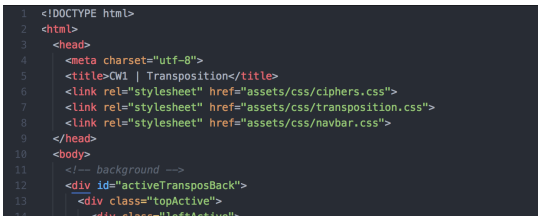

Figure 4: Head structure of every HTML file

When it comes to the body of the HTML, every page follows the same structure, see Figure 5. There are 4 div tags placed directly in the body. First, containing elements for the site's active background. This is the only div directly in the body that has an id assigned, just so it can be easily targeted in the JavaScript file, since it has to be updated with every

window re-size. The CSS pushes this div to the background by assigning a negative z-index to it.

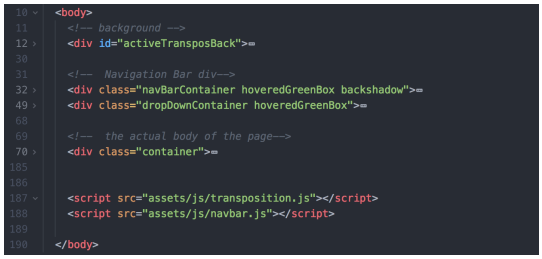

Figure 5: Body structure of every HTML file

Next, two divs include the whole navigation bar. One is the very container of the navbar, while the second is the drop-down menu. If there were more drop-down menus, each would need a new div tag.
In the code snipped below, we can see the classes assigned to the navbar. These are a good example of how combining style sheets (and therefore overriding styles) give nice results.

Listing 1: div tag of navbar in HTML

```
1  <div class="navBarContainer hoveredGreenBox backshadow">
2
```

The "navBarContainer" class belongs to the navbar.css and styles the size, content, and positioning. "hoveredGreenBox" comes from the transpos.css file that specifies the theme colour of the transposition cipher, while the "backshadow" is part of ciphers.css which is communal to all pages. This combining results in being able to have different coloured navbars that behave the same way, with zero duplicate code. See Figure 6.

Figure 6: Comparison of all different navigation bars

The last div contains all the actual content of the page. As can be seen in Figure 7, all of this HTML combines classes in a similar way to the navigation bar. Finally, all necessary scripts (JavaScript) files are linked to. It is crucial that this is done at the bottom of the body. The JS contains DOM references to objects in the HTML. If these were called before they appear in the HTML, the site would simply not have its functionality.

Figure 7: Variety of class combinations in div tags

3.1 Cipher algorithms

For each cipher, as mentioned previously, a standalone JavaScript file contains all the logic. Every cipher then has two methods that are most important - updateCipherOutput() and updatePlaintextOutput(). The naming gives away their function. They take in the input of the current textarea we are working with, and are called to use the other textarea as an output. The logic of the ciphering and deciphering (respectively) is contained in these methods.

To go through all ciphers would be timely and redundant, so let us examine the enciphering method of the transposition cipher as an example.

Listing 2: Method of enciphering text using the transposition cipher

```
1  function updateCypherOutput(){
2    var input = inputPlain.value;
3    var numOfRows = Math.ceil(input.length/numOfColumns);
4    var plainTextArray =[];
5    var cypheredString ="";
6
7    for (var i =0; i<numOfRows; i++){
8      var oneLine ="";
9      oneLine = input.slice(i*numOfColumns, (i+1)*numOfColumns);
10     plainTextArray.push(oneLine);
11   }
12   for(var i = 0; i<numOfColumns; i++){
13     for(var j = 0; j<numOfRows; j++){
14       if(plainTextArray[j].charAt(i) === "") {
15         cypheredString += specialSymbol;
16       } else {
17         cypheredString += plainTextArray[j].charAt(i);
18       }
19     }
20   }
21   inputCypher.value = cypheredString;
22 }
23
```

To encipher in transposition, we imagine placing the text in a grid, where numOfRows is calculated as the length of the string divided by numOfColumns - this is decided by the user's chosen key. Then, we slice this string into rows and push each individual row into an array plainTextArray. We also create a new empty string cipheredString. By iterating through a loop within a loop, for every row in every column, we add the current letter into the cipheredString. In other words, we create a string that reads the grid from top to bottom, as opposed to left-to-right. Finally, the textarea inputCypher is used to view the ciphered string.

3.2 Design and Styles

The final design of the web-pages is considerably close to the initial sketches and ideas. Screen-shots of the finished product can be seen in Figures 8 through 12. When designing the sites, the basic UX principles [2] were kept in mind. All pages are consistent, desirable, straightforward to use, and created with attention to detail.
For instance, the textareas on cipher pages serve both as an input and output. To indicate to the user what is being translated, the arrows between the two inputs light up in the direction of the ciphering. Also, elements that serve as links usually change in colour when hovered over, in order to indicate interactivity.
Cipher Madness makes use of simple animations (eg. content jumping up on page load, clickable elements darkening when

hovered over), which are useful in the aesthetic department. Elements changing state through a transition as opposed to just switching from one to another, makes the website feel more dynamic and responsive [3] - exactly what a modern user expects it to be.

4 Critical Evaluation

4.1 Functionality

To ensure the best results in functionality, the pages underwent a fair amount of testing. Mainly the ciphers, as they contained the most algorithms, were prone to bugs and unexpected results. Mostly these bugs were an easy fix, however, some improvements could always be done. For example, the binary cipher will not recognise a character if they're not separated. "1000001" should translate to "A", but this is only the case when the input is followed by a space: "1000001 ". One way of fixing this would be only enciphering letters of the alphabet, but that would also add a different kind of restriction. It is therefore down to personal choice and what the user would consider less of an inconvenience.

Possible Improvements
All testing and development were targeted at Chrome. To ensure higher functionality, better support for other browsers could have been considered. After launching the project in Safari, it became clear that should have been part of the process from the very beginning, since a few elements do not behave as expected outside Chrome and the performance is considerably lower.

4.2 User experience

The website is responsive and does what is expected from the user's perspective. It's important that the same user experience is provided in any format, considering the vast number of screen sizes it could be viewed on. Therefore, the layout responds well to different window sizes on a computer, while keeping the content centered and in sight.
It is also responsive in regard to efficiency. For instance, the background of the Binary cipher, is not static. It is a mesh of alternating 1's and 0's. For this to be responsive to screen size and to ensure that the mesh fills out the whole page in every occasion, the element needs to be populated every time the screen is resized. There were several ways to do this with JavaScript. One way is to change the innerHTML of the background div and add, for example, 1 n-times. This worked but changing the innerHTML retrieves the whole content of the tag before being able to add anything. Instead, the appendChild() method has been used to pass in a span - much more effective with minimal lag. [4]
Many similar considerations had to be evaluated to ensure an optimal user experience, even for a project of this size.

Possible Improvements
Regardless, the responsiveness could be improved by using a library such as Bootstrap or implementing the same kind of logic that Bootstrap provides for formatting. This would make the website mobile-friendly and provide a more responsive layout.
A prime example would be the very home page, where three horizontally placed blocks scale down when the window shrinks. Instead, they could re-position to a vertical formation before down-scaling. This is not necessary, but would ensure the elements are more easily readable.

4.3 Advanced features

The features demonstrated in this project certainly exceed the materials and exercises of the practical classes of the module. With implementing drop-down menus, a custom navigation bar, JavaScript formatted interactive backgrounds, and responsive design an attempt has been made to create a professional-looking, functional web-site. It has also been published via GitHub Pages, and can be publicly accessed at this link

5 Personal Evaluation

There are many takeaways from this project. Developing the website provided a much deeper understanding of how web technologies are interconnected and play together to be interpreted by the browser.

It has become clear that organisation within the code is absolutely crucial in web development. Messy code and a sloppy file hierarchy can grow into an unreadable mess very quickly, so it was important to think ahead. This didn't only apply to CSS, but JavaScript as well. A lot of bug fixing was done due to the DOM references often affecting more objects than intended. And the very same applied to CSS.

In fact, most time was spent with trying to get the CSS right. This is only reasonable since the whole project is so heavily focused on delivering a nice user experience. At first, the styling seemed very straightforward, which it actually is, but there's more to it than just applying colour to elements. As the project got more complex, and each .html file was using more than one CSS style sheets, suddenly the specificity of selectors became a concern and some changes had to be made along the way.
Other problems occurred when implementing the ciphers. Even though these were just simple bugs and unexpected behaviour that needed fixed, they managed to cause some head scratching and at times slowed the workflow down considerably.

The website has been shown to several people in order to gain some usability feedback. [5] Observing how these people used the website and what assumptions they made, some changes have been made to the design. For instance, the "More info" on the cipher pages changes colour to indicate it's a clickable button. This was not the case initially, and seemed to be a usability concern.

Part of this coursework were also directed studies. These included sources like MDN, w3schools, and stackoverflow, however, the one that needs to be addressed the most is the Web Developer Bootcamp course by Colt Steele (purchased on Udemy). The first half of this course (about 26 hours of video - excluding additional exercises) helped understanding the basics of front-end web development and how HTML, CSS, and JavaScript work on a fundamental level. Without

doing this course, the project might have been even more time consuming.

Conclusion
In overall, it is fair to say that a very solid attempt has been made. Even though not perfect, the user experience is very pleasant, the website is nice to look at, and the responsiveness is on a good level (considering no libraries were used). The implementation of the ciphers seems to be bug free at this point as well. *Personally, I feel this project has been a success and definitely helped a lot with developing my skills.*

References

[1] dCode, "Trithemius ave maria," 2018.

[2] M. Soegaard, "Usability: A part of the user experience," *The Interaction Design Foundation ApS*, Jan. 2018.

[3] C. Steele, "The web developer bootcamp," Jan. 2018.

[4] S. Allen, "Javascript innerhtml versus appendchild performance," 2007.

[5] T. Churm, "An introduction to website usability testing," *Usability Geek*, July 2012.

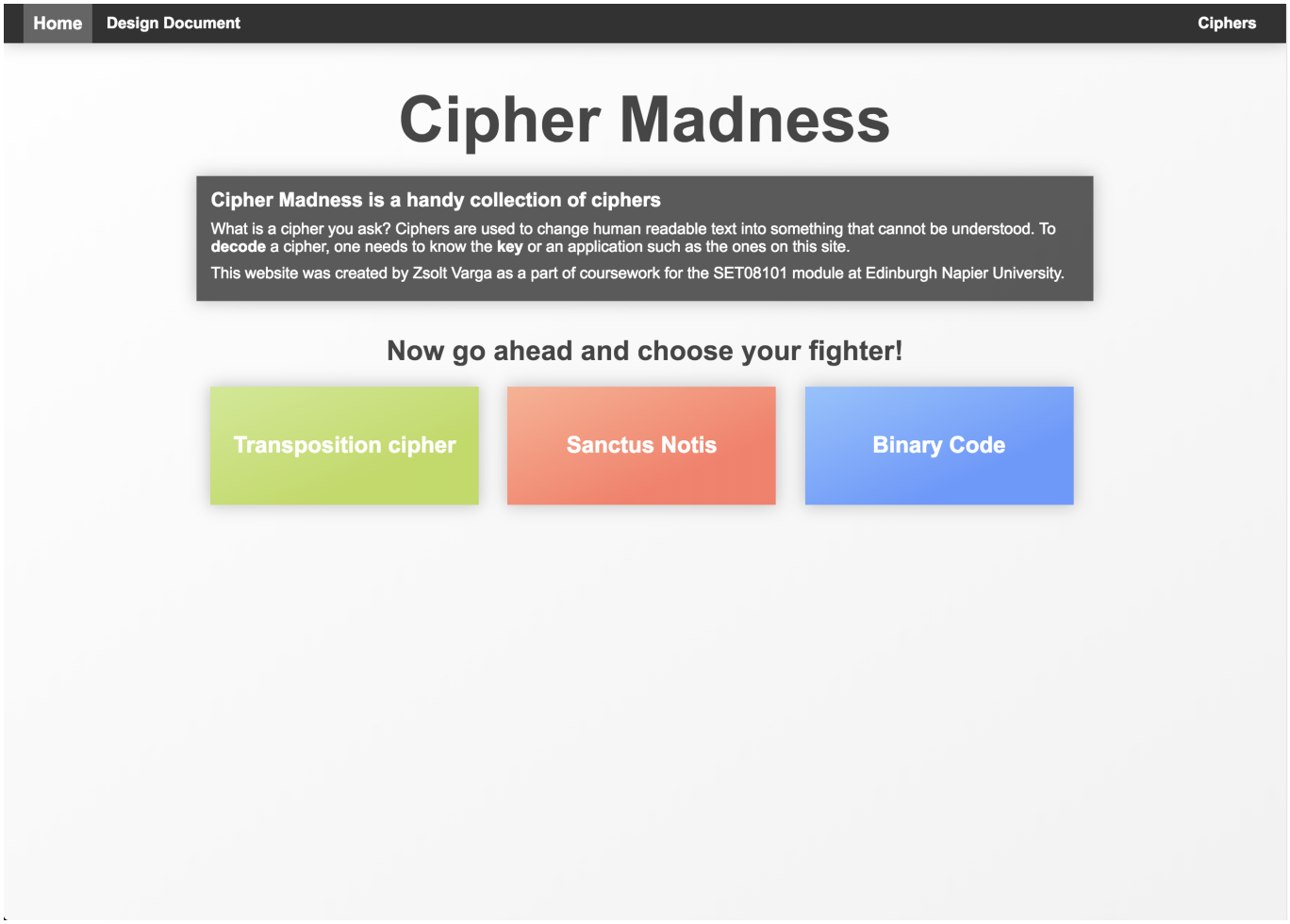

Figure 8: Screenshot of final Home page

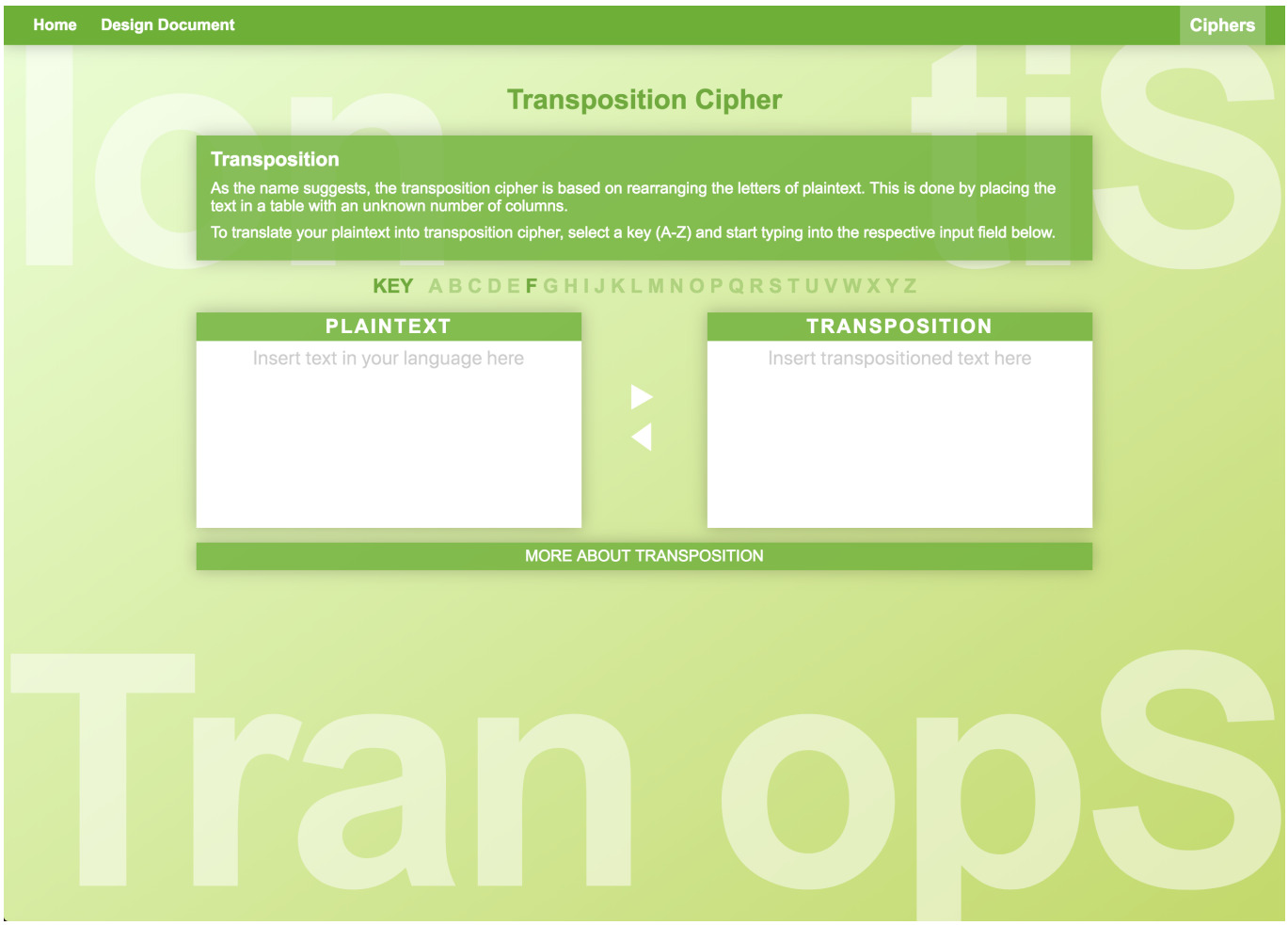

Figure 9: Screenshot of final Transposition cipher page

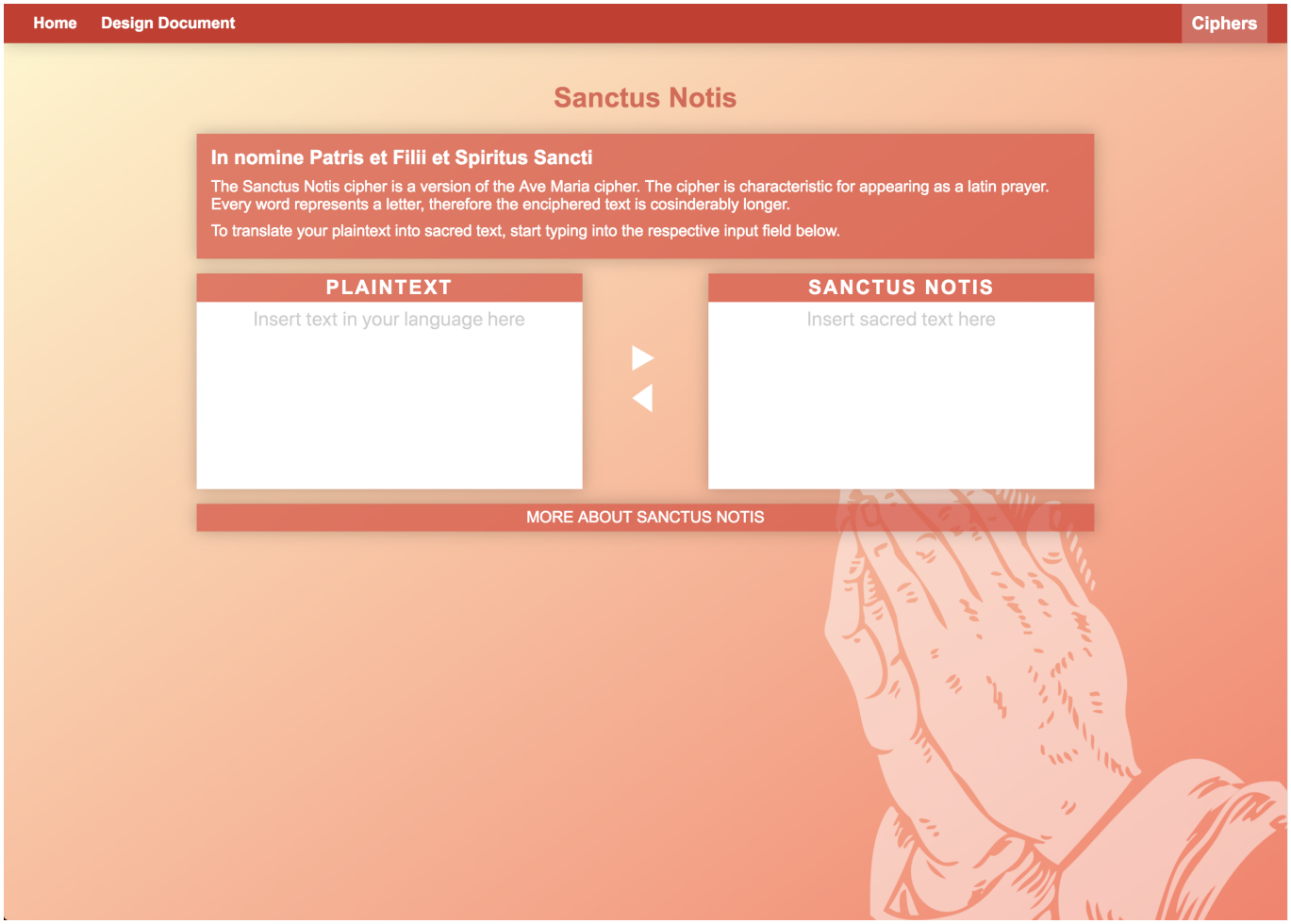

Figure 10: Screenshot of final Sanctus Notis cipher page

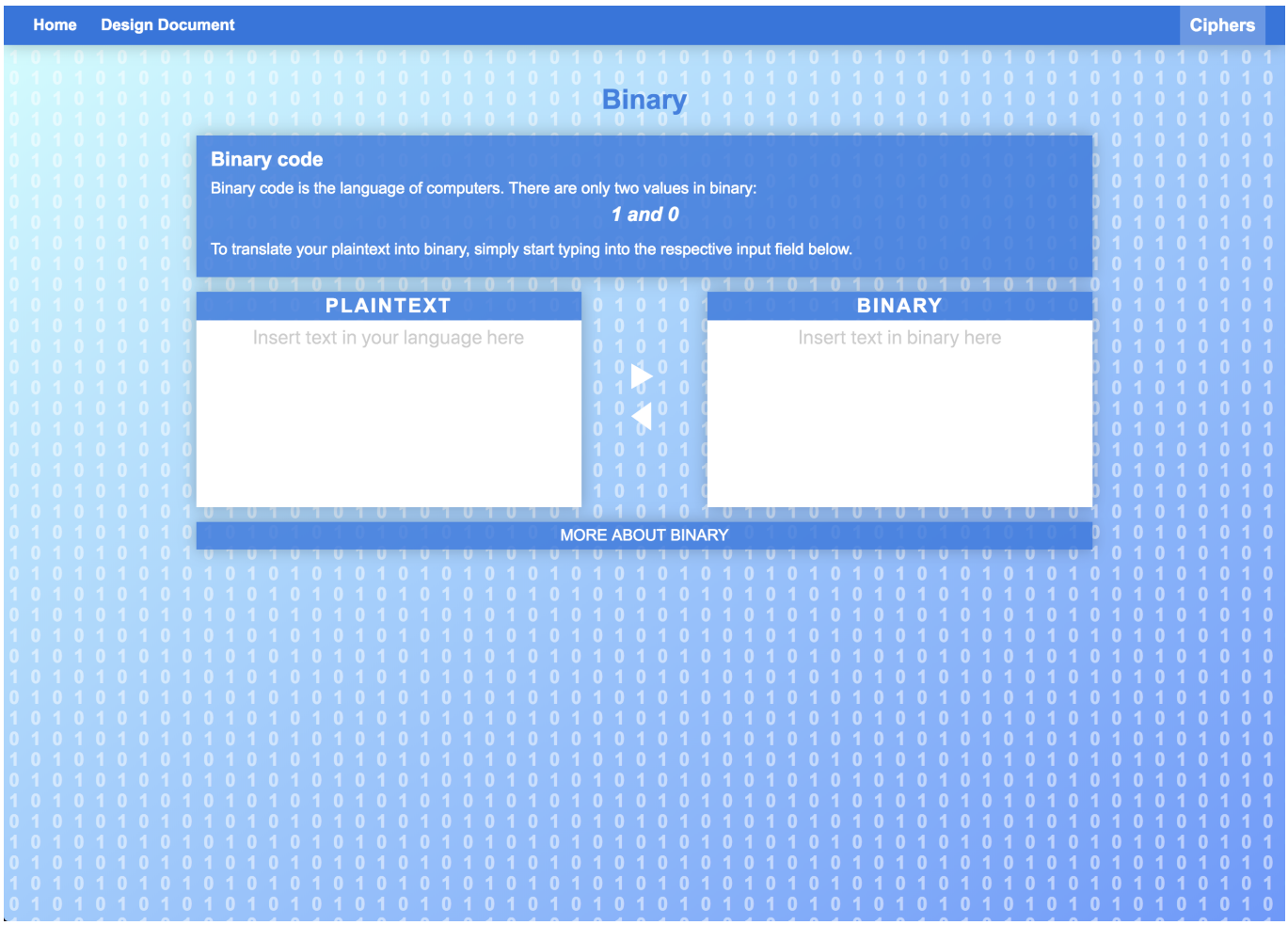

Figure 11: Screenshot of final Binary cipher page

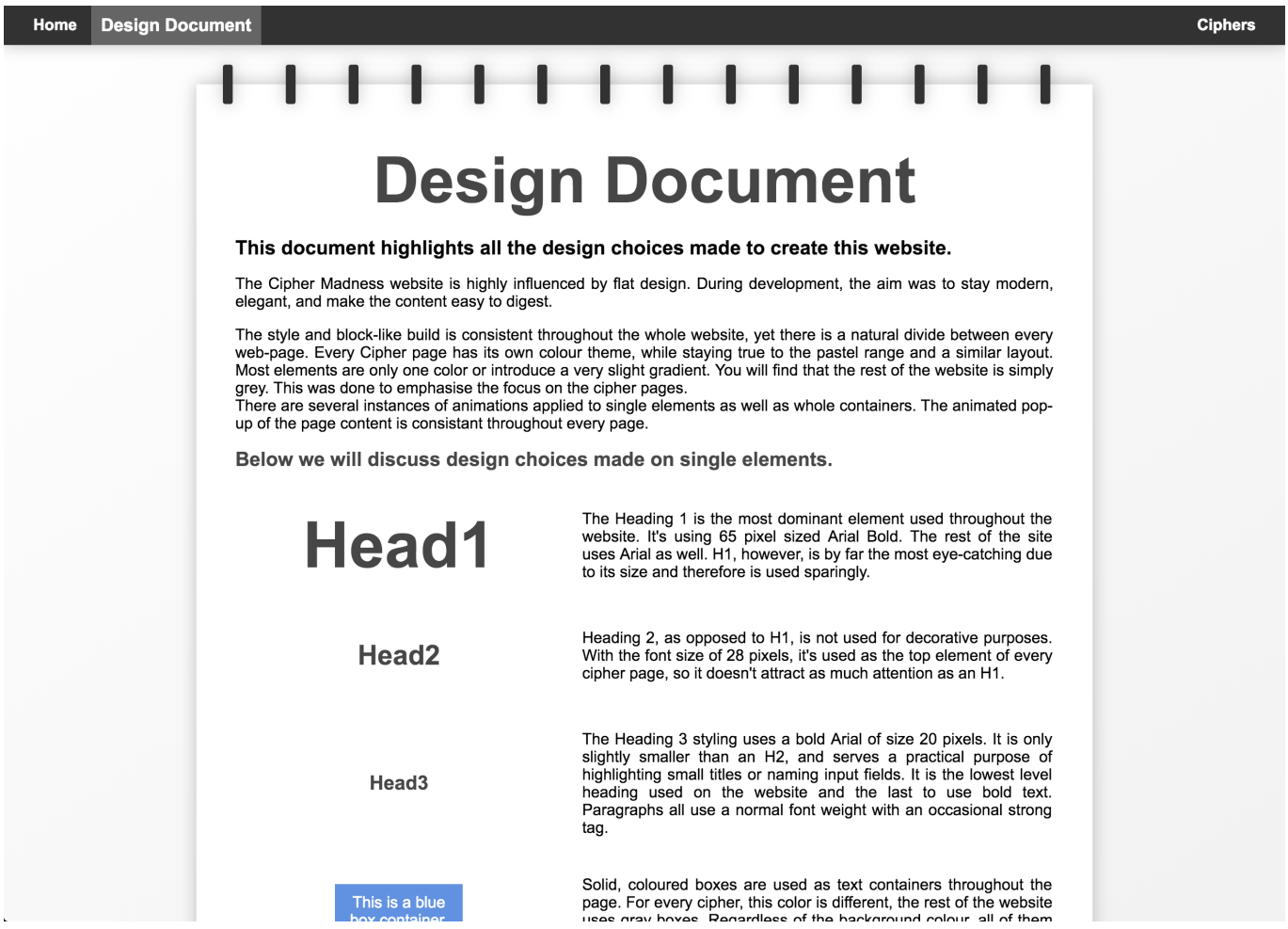

Figure 12: Screenshot of final Design Document page