EncryptionJS

Course number, May 9th 2017

# Completed individually by Eric Hughes, 1412844

## Phases completed (circle all that apply): 0 1 2 3 4 5

// 1 small bug in IE8 after decryption you cant copy the values from the container has todo with compatability issues. Seems to work fine on VM but has trouble on emulator but encryption does work.

## Overview of Implementation

Build view first from the designs I saw on online such as mozilla emoji encryption and other forms in general.

In general for the web design courses I took before starting that the 3 column layout seems to be a very infamous design pattern that most developers follow.

I decided to make a small mockup in illustrator that gave me an idea of what to tackle first instead of design as I code which I found does not work as well.

I then began building my view this way having all my keys in the middle and the before and after so reading from left to right was a simple process.

The biggest constraint for the view was making sure the classes were designed appropriately to match the css rules. What I mean is that css is only applied as necessary and there are no redundant css rules being applied.

I next started doing some reading on the emoji notes before even starting to code my js file. Wanted to understand what Maya was talking about, why theres so many libraries to make polyfills for emojis why they behave so weird etc.

Decided to use the emoji.codePoint comparison as my convention instead of comparing the emojis “as is” that way I know the equality always will work.

The following thing I wanted to implement is getting the value of the textbox and working with it in js.

I then began building my emoji and alphanum arrays and began testing how they work with one another. For example I would see how strings are different in js then in java/c#.

I ended up creating a custom emojiToArray method that took emojis and converted each emoji into an array element since the research I did on emojis said that the length property is 2 even though its 1 character long.

I then implemented encryption using emoji only before regular ceasar cipher with alpha numeric which was a mistake that I learned the hard way.

I should of build regular ceasar cypher and then added in an extra check to see if the key is of type emoj, then just parse it to normal and continue my logic. I did the reverse, I had to implement regular ceasar cipher into an emoji encryption logic which has checks everywhere what type the key is.

Once encryption was fixed I stated to look how to check if you are in internet explorer 8. So I installed a VM and began debugging for internet explorer 8 which was a nightmare but ended up working after 4-5hours.

The next thing I created was the wizard, and since I had experience from the web design courses I managed to create the wizard rather fast but still took another 3-5 hours.

Once the wizard got to working I then implemented the weather data api encryption. After a couple of tutorials online I managed to figure out the syntax and it ended up working quite well.

I then implemented the cookie feature of the last city they entered and if they visited for the first time. The cookie implementation was way easier than the weather API.

I think noticed that my encryption was not working that well when the key was greater than the length of the alphanum array. I then spent 3-5 hours debugging my code and finally found a solution that works very well.

Now began refactoring, testing on all browsers, I then saw in the specs that a lot of the marks are on decomposed functions into smaller functions instead of do it all functions.

## Known Issues or Areas to Improve

The biggest bug I found is coding in IE8 and checking what type of key you are given. I think I tried to do have encrypt and decrypt in 2 methods and did no spend enough time in my design of how I want to do my checks instead of just doing it as fast as possible.

I think the coding environment in IE8 is bad and very unintuitive to code in.

The logic behind the Cesar cipher should be done with test cases and on paper instead of doing it all in your head.

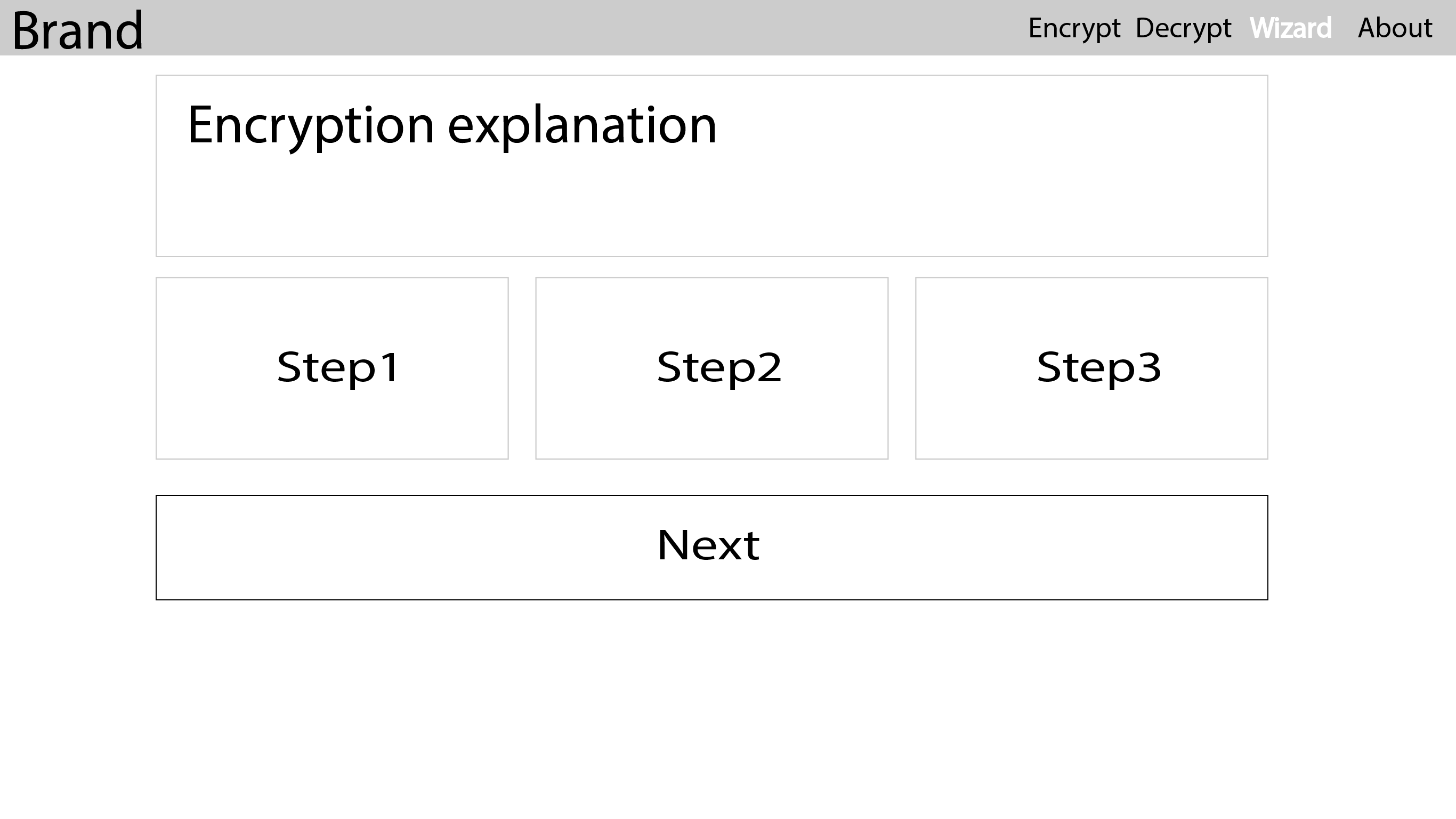
List 20 things you need to test manually in your final app to check that the features work.

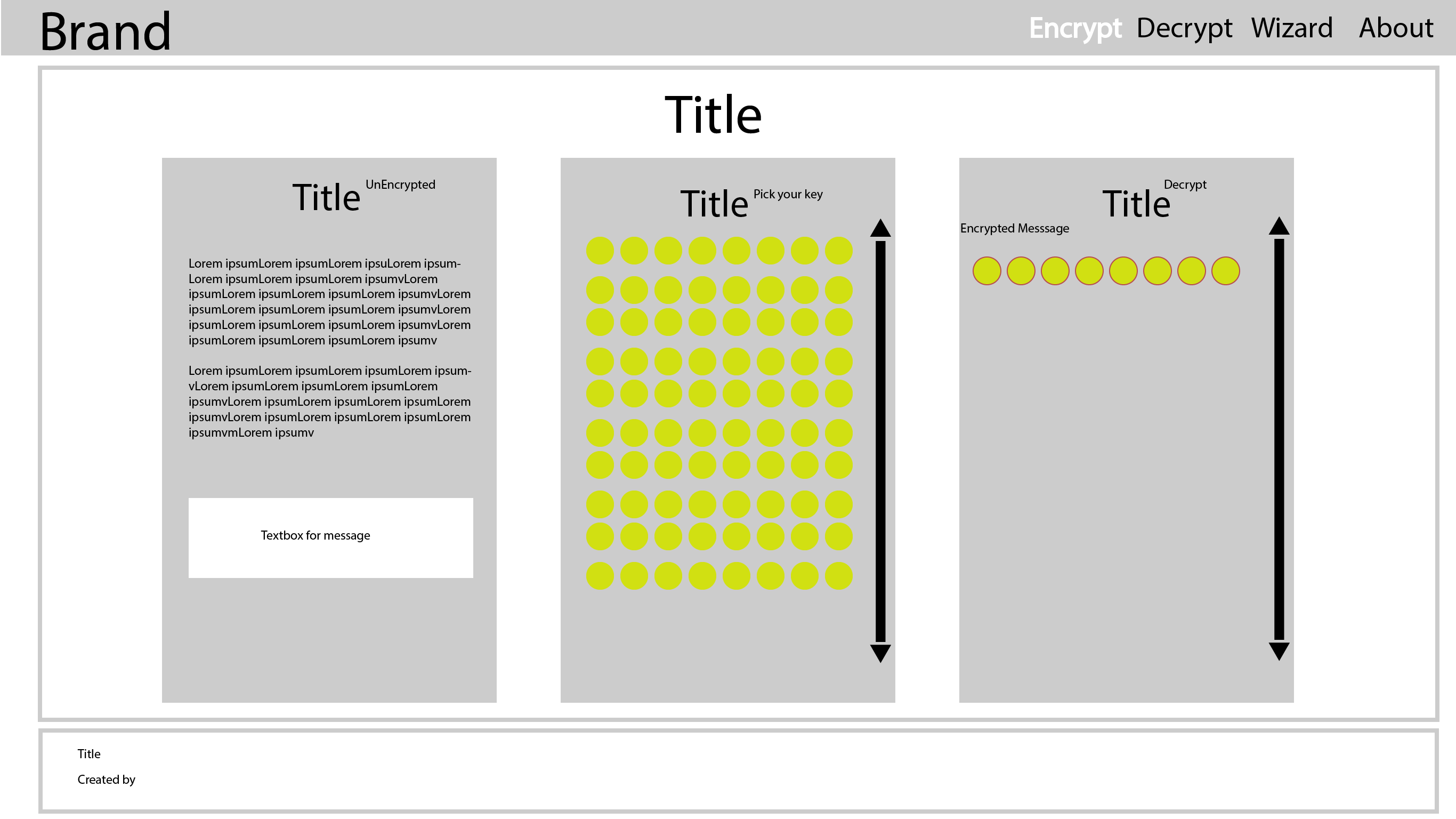
Include cover page of your on-paper submission.

1. Check width distribution when re-sizing the window on browsers
2. Make sure no redundant css properties are being applied
3. Validate user input when he/she enters regular English message
4. Make sure characters from message are matched in alphanumeric array
5. Make sure that user can only click keys to change the text and not create their “own key”
6. Make sure that the key being selected is matched up with the proper value in the emoji or alphanumeric array
7. Make sure wizard works across all browsers
8. Test that encrypted message has just as many characters as the original non encrypted message
9. Make sure weather encryption works both with any city they provide that is serviced by the API
10. Make sure that the resulting emoji displayed on screen represents the weather data retrieved from API (sunny weather means sunny emoji)
11. Decrypting is same procedure but in reverse, make sure user can copy paste encrypted message into decrypt box in decrypt html page
12. Make sure key they choose matches up with the key being displayed on screen
13. Make sure that if they choose a key greater than the length of the English array that it loops over the length using modulus and calculates the right displacement to uncrypt message
14. Make sure if user uses IE8 or older that regular cesar cypher is displayed over emoji
15. Make sure user cannot mix and match regular English and emoji characters for encrypt and decrypt
16. If weather API is down notify the user that they cannot encrypt their data using weather data
17. IE (older browser) make sure correct array is being used for encryption and decryption and keys
18. Make sure that wizard is displayed if the user has never visited website (check if cookie is present)
19. Make sure that the key is not able to be seen when inspecting element such as “selected” attribute inside the src of the array elements
20. Make sure that the user can combine keys with messages appropriately (first word has key 3) second word has key 7

## Mockups Screenshots

[ e.g. encryption view and wizard in wide screen in Edge, narrow screen in Firefox]







Final Product Screenshots

