Student number: 2410516

University of Stirling, computer science

CSCU9N5 assignment

Report

# TABLE OF CONTENTS:

# Product Description

## What is it?

ManxVisual, also abbreviated as MV. ManxVisual is a web application that enforces a different learning style for the basics of programming logic. The idea behind MV is to allow people to learn the basics of how blocks of code such as ‘if statements’ work in a programming language. MV is designed with real life programs in mind, with the terminal being the main area of where the code is executed and displayed.

The twist is that MV displays the terminal similarly in a Mac OS (Operating System) whilst also showing another section that allows the end-user to understand how the blocks of statements are executed. A section of the page is assigned to be the ‘Method to be executed’. This method mimics method from any programming language but it’s displayed visually instead of using text like in an IDE such as Visual Studio.

The components, also known as the blocks of code, are taken from the ‘Components’ section and then dragged into the ‘Method’ section to allow the user to build a custom method. This can then be ‘Run’ and then the output is displayed in the ‘Terminal’.

When I was brainstorming the ideas behind what web application to make, I decided to go for something that would provide an intro to programming without actually writing any code. The problem with teaching programming is that it’s hard to cater for all needs. Some people learn from just deep-diving in; others by learning from the basics with pieces of texts or instructional videos. Then, there’s those that learn visually and need to learn from the nitty-gritty stuff. ManxVisual teaches the very basics of programming logic in a visual but smart way.

## Who is it aimed at?

ManxVisual is aimed at those who want to get into programming but don’t know where to start. It uses visual methods to teach whilst maintaining some text on the screen for explanation purposes.

The age range is any age of 10 and above. Those who love to learn visually are the ideal target due to the dragging and dropping of components involved in the web application.

To be critical, MV could have been used by ages 4 to 10 too but this is only aged 10 and up due to the text potentially containing vocabulary that is out of the standard children’s vocabulary set. Anyone wanting to pursue programming but knows nothing, would find this web application extremely beneficial to grasp the concepts of blocks of code such as ‘if statements’ and ‘for loops’.

## How would it be delivered?

The web application would be delivered via two possible methods. Due to no specifics being outlined in the specification, I will describe two different methods that I could possible use to deliver the web application.

Method one:

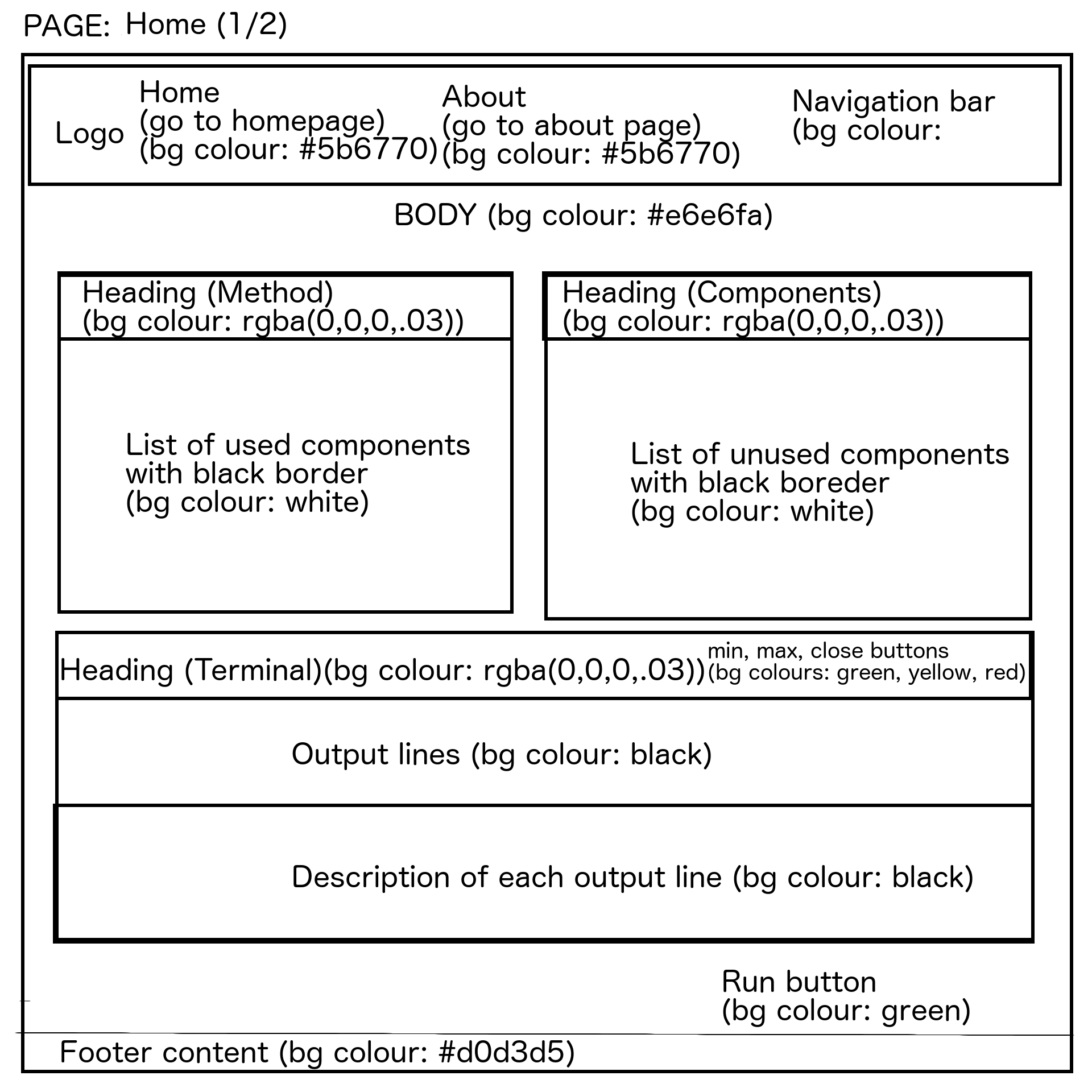
This involves compiling the source code and then transferring it into a folder. The source code within the folder would be deployable straight away with the client having to solely copy and paste the source code straight into their web server. The folder would then be zipped up and compressed, finally sent via email.

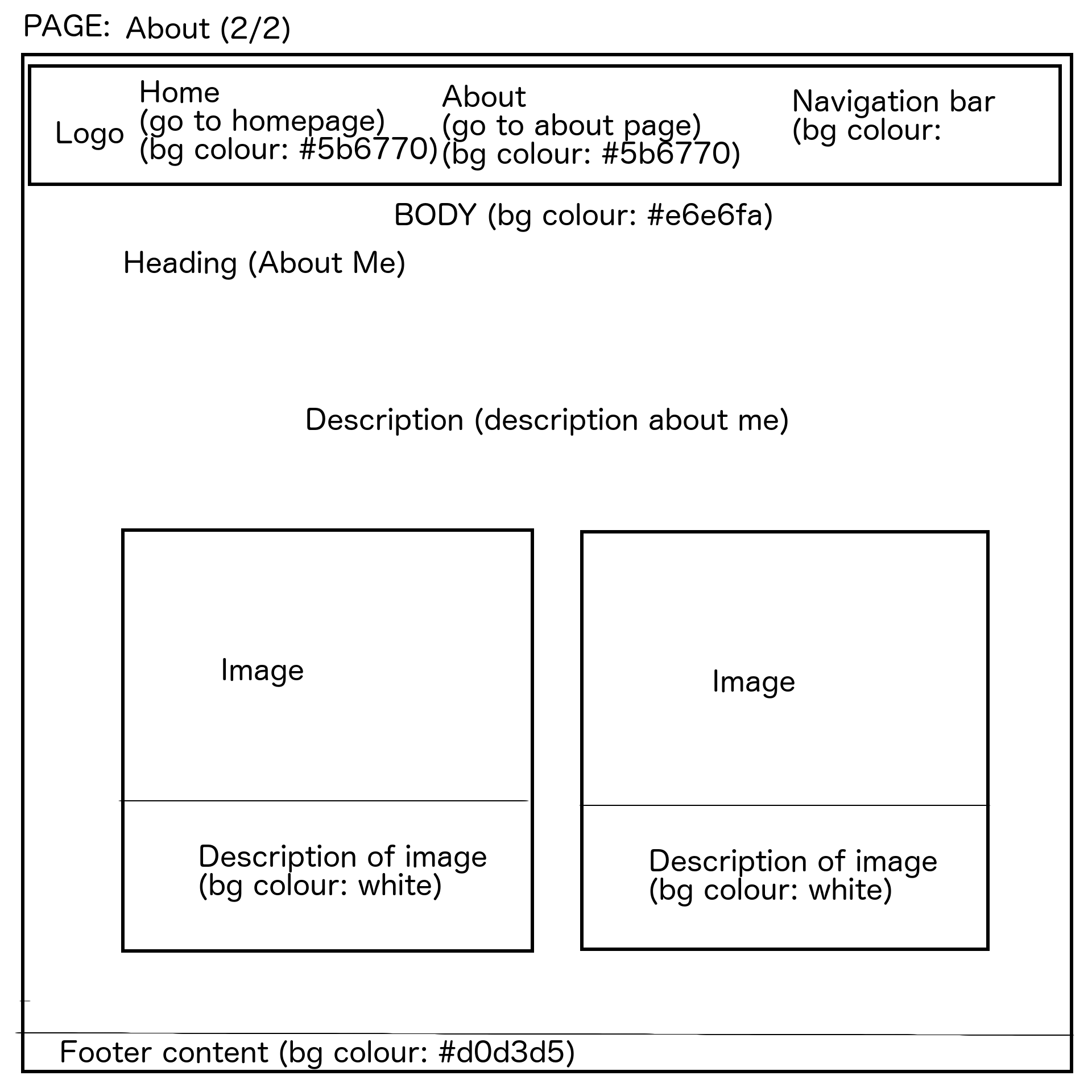
Method two:

This method involves deploying the web application onto the client’s web server. This would involve signing a contract with the client to handover any of the server credentials. I would then compile the source code and upload the web application via FTP onto the client’s web server.

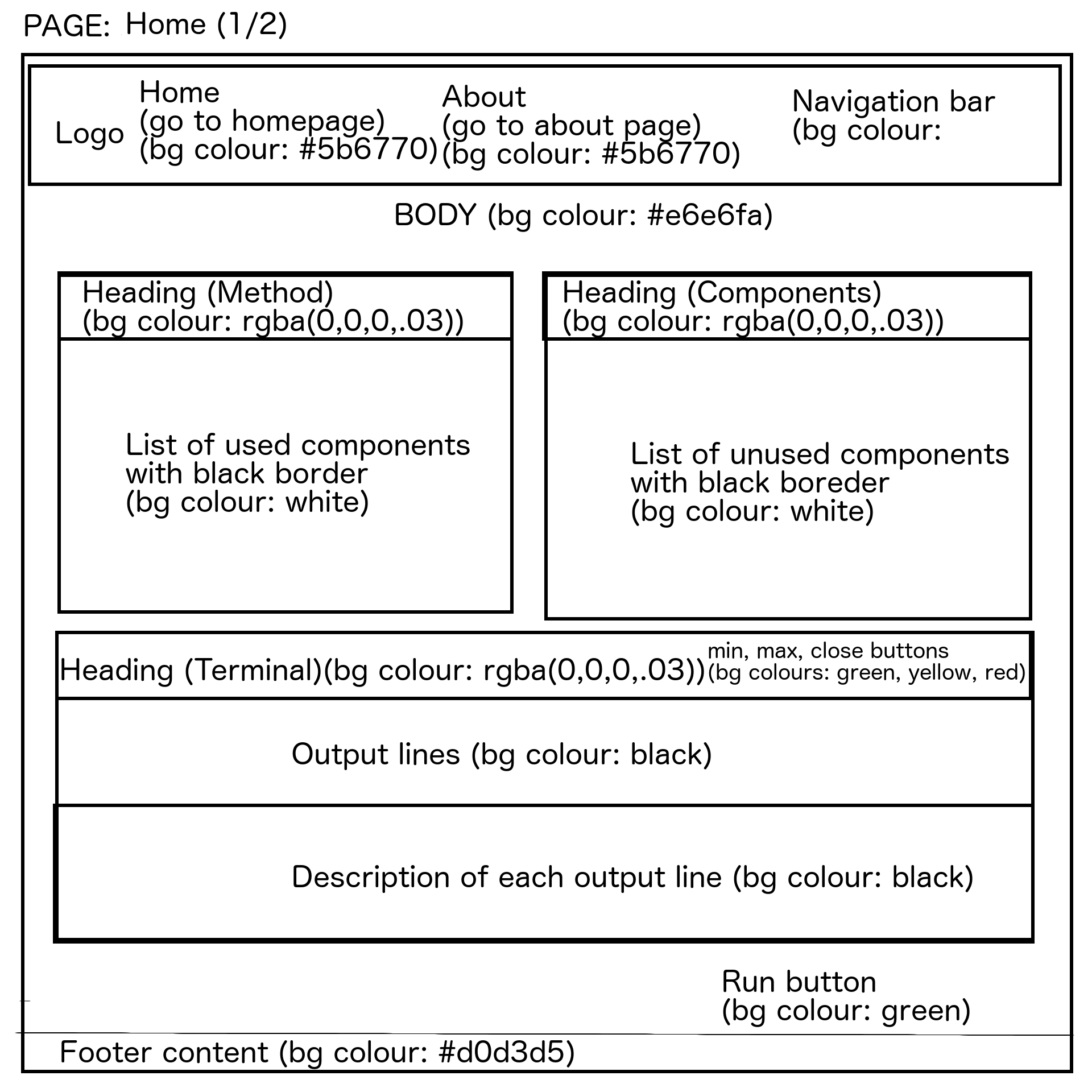
# Design

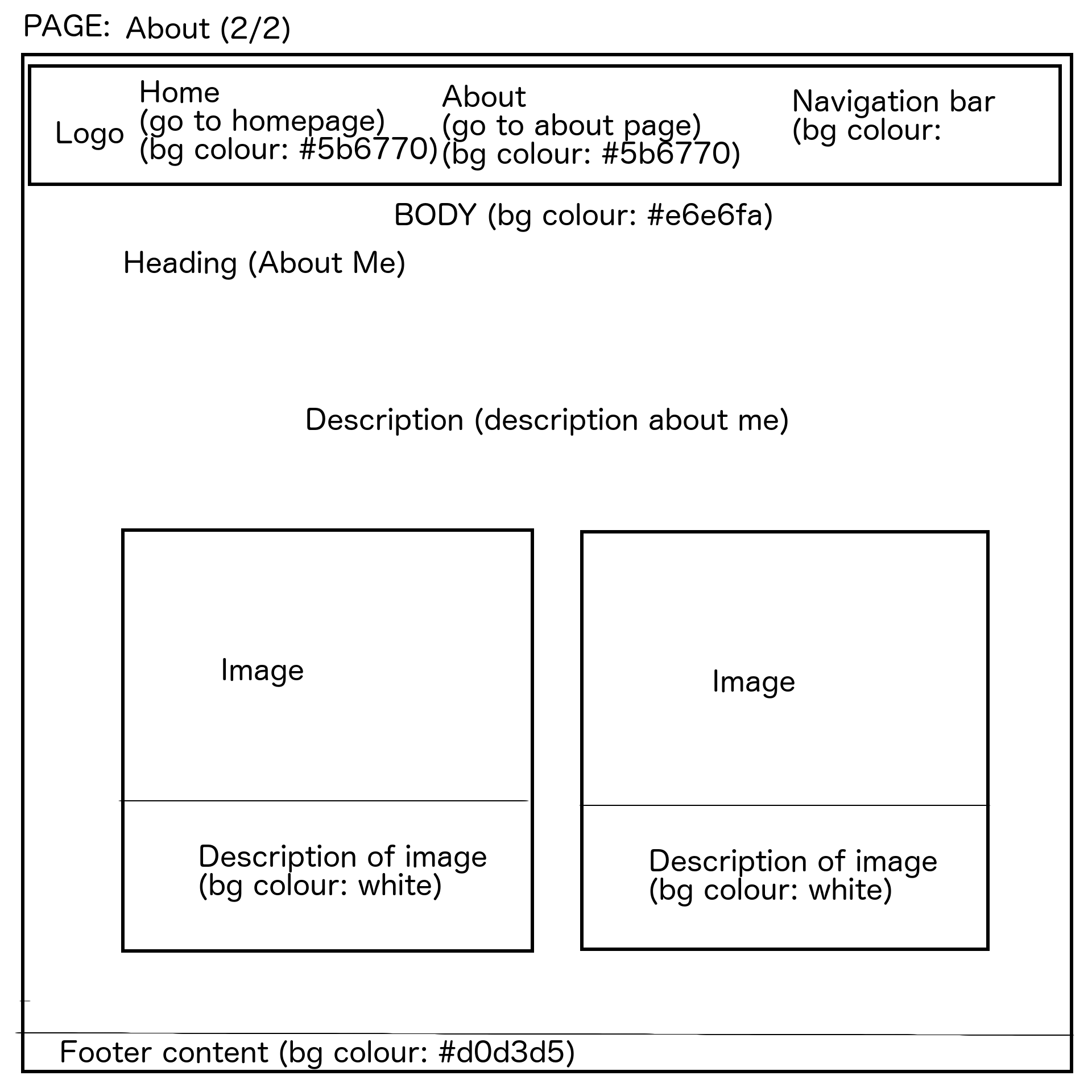
## Storyboard:





## Navigation Map:





## Task Analysis:

## Design Decisions:

# Prototype Description

## What does the prototype contain and why?

The prototype contains most of the full application but allows the client to see how the application will fully function once it is complete. It allows the client to use the main functionality of the web application, this is done by allowing the client to move components (blocks of code) over to the method section to allow execution of the method.

Once a component is moved over to the method section, the component block will transform into a block with input elements. The components available in the prototype are blocks of code that allows execution of ‘if’ statements and ‘for loops’ too.

The input elements for the ‘if statement’ block contains a few input boxes to allow two Boolean values whilst also allowing the user to type in an output for the block of code.

The input elements for the ‘for loop’ block contains some input boxes to allow the user to enter in integer values to set values such as the starting index, ending index and the increment value for the for loop to be executed.

Once the user has inputted the values of the blocks within the method section, the user can scroll down to press run. This allows the user to run the method and display the output as it would do in a normal terminal whilst also explaining how the blocks of code work in hindsight.

The prototype allows the user to know more about me as a person and why I’ve created the program by going to the About section which is clearly shown in the navigation bar at the top of the website.

There are animations here and there on the website. One of the animations is the orange loading indicator, located in the navigation bar, that moves when something needs to be loaded such as the initial website load at the beginning when the user opens the website.

Another few animation triggers are located within the terminal window area. The close, minimise and maximise buttons all similarly mimic how those actions would be carried out on a Mac OS (Operating System).

Finally, the prototype includes a very short tutorial video when the user first loads the website on the homepage. This tutorial video is short but effective in explaining the sections of the page.

## How does it differ from the full version?

The main difference between the prototype and the full version is the number of components available for the user to utilise. In the full version, I would include more components to mimic other blocks of code such as assigning a variable.

In addition, I would also allow the user to add more methods to the program that is created by them visually. This would allow the user to understand concepts such as scopes of variables for example.

## What would also be sent along with the prototype to the client?

The client would get the prototype but would also receive a handbook explaining how to deploy the web application onto their server. The handbook will also contain instructions on how to use the web application whilst also containing a page on contact detail in case there are any bugs found within the application.

# Usability Testing

# Web Technology References

Below is a list of web technologies used for this project:

(Format: [INSERT NUMBER]: TECHNOLOGY NAME ([INSERT LINK TO TECHNOLOGY]) – [INSERT REASON FOR USE])

1. Angular 4 (<https://angular.io/>) – Angular4 is the base of the project. This library has helped build the structure of the project by extending HTML's syntax to express my application's components clearly and succinctly. It also provided routing to change pages of the website very easily without having to refresh the page – this used the MVC design pattern to create the web application.
2. Bootstrap (<http://getbootstrap.com/>) – I used Bootstrap to speed up development by using ready-made classes to style components and also lay out the pages of my web application.
3. Angular SortableJS (<https://github.com/SortableJS/angular-sortablejs>) – This library allowed me to create the functionality of easily dragging and dropping the components from one section of the home page onto another section of the page, in turn allowing sorting of elements on the page.
4. JQuery (<https://github.com/jquery/jquery>) – JQuery has been used to allow me to use other libraries such as Angular4 and Bootstrap. It also allowed me to write cleaner and less code.
5. SortableJS (<https://github.com/RubaXa/Sortable>) – SortableJS has been used to support the Angular SortableJS library above.