Student number: 2410516

University of Stirling, computer science

CSCU9N5 assignment

Report

Table of Contents:

[Product Description 3](#_Toc499537219)

[What is it? 3](#_Toc499537220)

[Who is it aimed at? 3](#_Toc499537221)

[How would it be delivered? 4](#_Toc499537222)

[Design 5](#_Toc499537223)

[Storyboard: 5](#_Toc499537224)

[Navigation Map: 7](#_Toc499537225)

[Task Analysis: 8](#_Toc499537226)

[Design Decisions: 8](#_Toc499537227)

[Prototype Description 10](#_Toc499537228)

[What does the prototype contain and why? 10](#_Toc499537229)

[How does it differ from the full version? 10](#_Toc499537230)

[What would also be sent along with the prototype to the client? 11](#_Toc499537231)

[Usability Testing 12](#_Toc499537232)

[Test: 12](#_Toc499537233)

[Comments after test: 12](#_Toc499537234)

[Web Technology References and Site Access 13](#_Toc499537235)

# 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’.

Let’s take two personas two be used throughout the report:

1. Persona One’s goal is to be able to input the values into an ‘if statement’ component within the method section.
2. Persona Two’s goal is to be able to use two components within the method section and then execute the method within the terminal.

## 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:

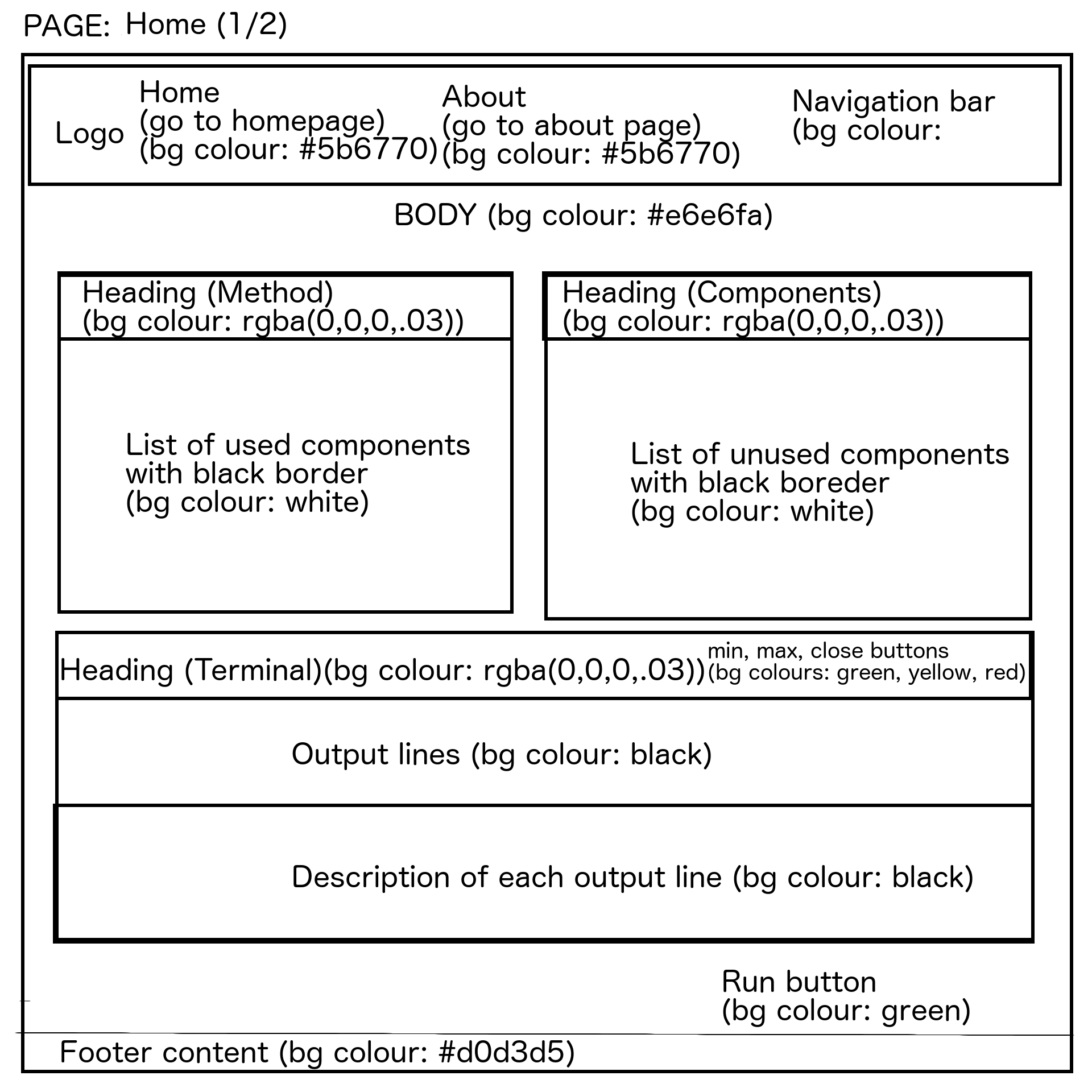
This involves compiling the source code and then transferring it into a folder. The source code within the folder would be deployable straight away with the client having to solely copy and paste the source code straight into their web server. The folder would then be zipped up and compressed, finally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

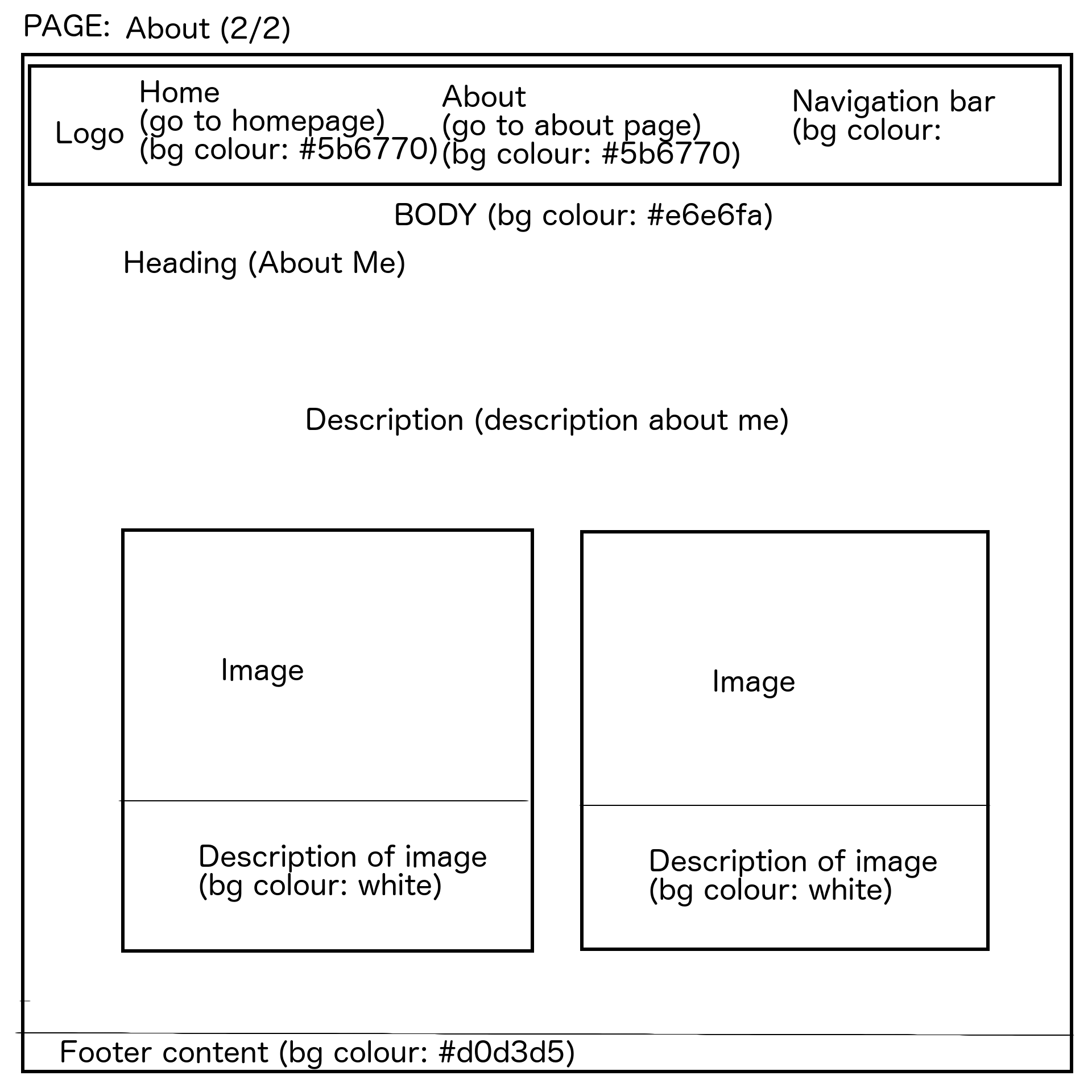
Method two:

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:





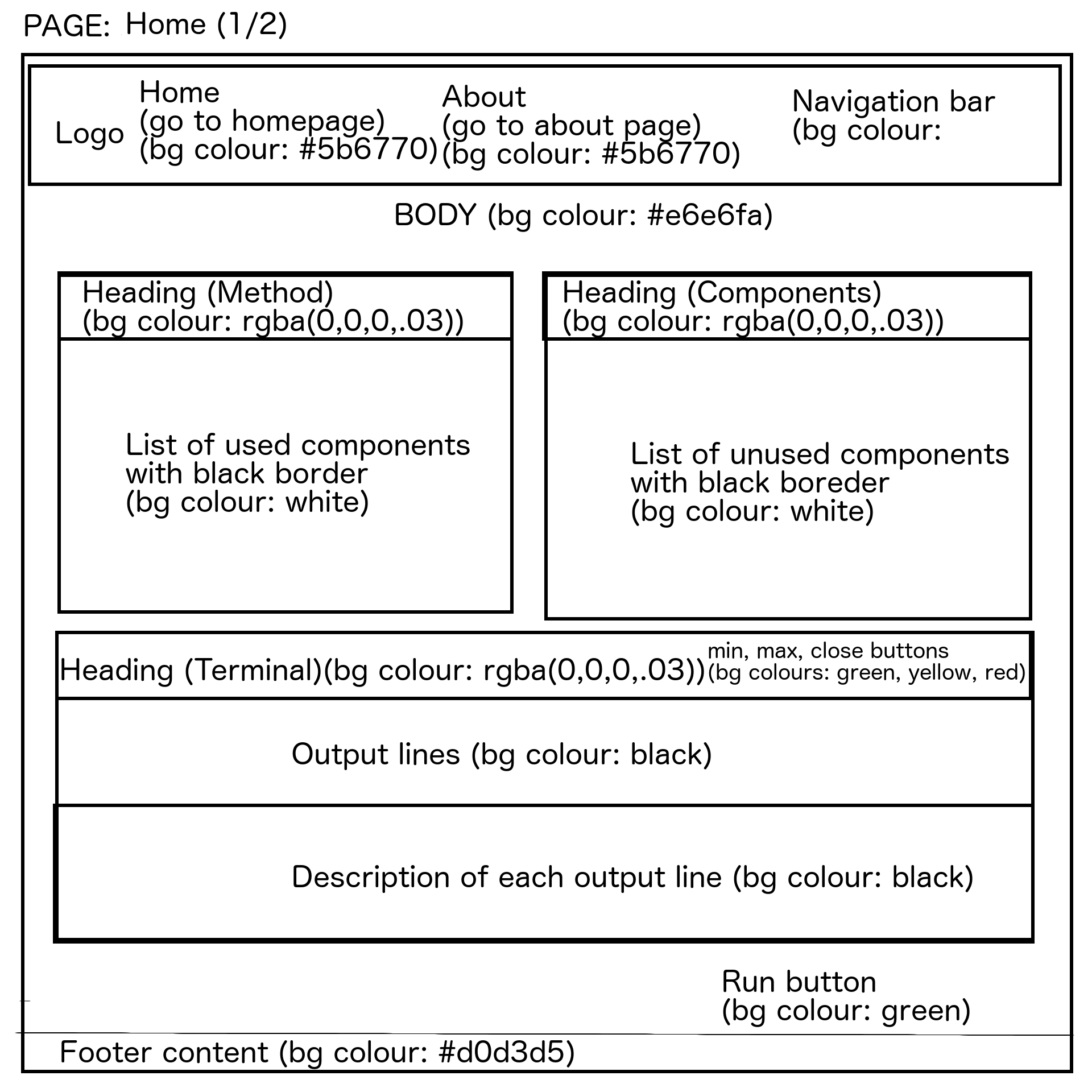
When creating the design for this project, I considered the types of users that would use this application on the internet. The best example is of ‘Persona Two’ where that persona should be able to use two components from the components section within the method section.

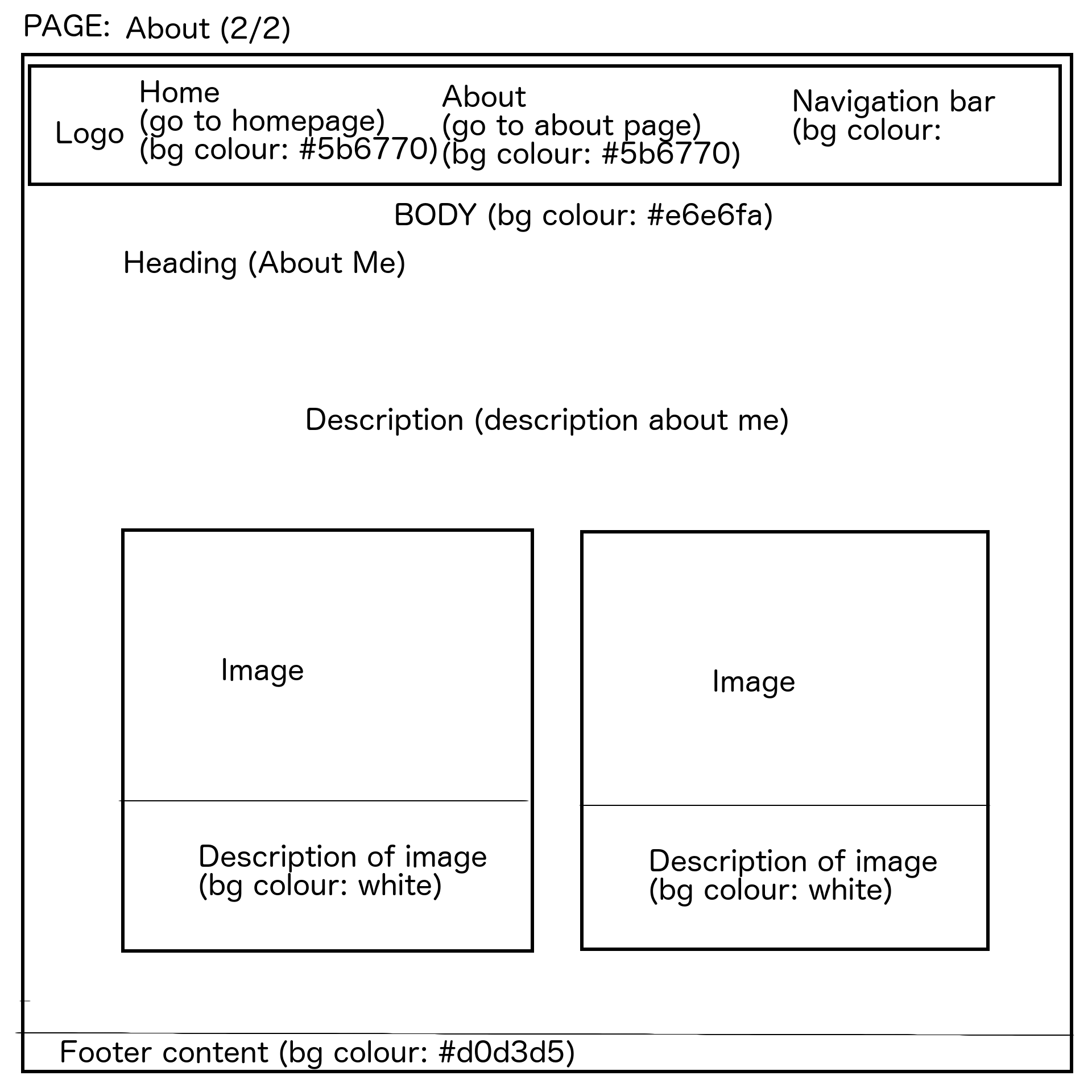
I took this into consideration and decided to split the page in half where one side is for unused components and the other is for the method to be executed and outputted to the terminal below. Users usually scroll through the website, downwards, to reveal more content such as the “Run” button that allows the user to run the method that they have created using the components.

After the user drags and drops components, then inputs the required parameters, the user automatically thinks what they need to do to run it – this is where they scroll downwards to find the button.

To be critical, there could be an overlay button whilst at the top of the page, to allow the user to go straight to the bottom of the page where the run button is located. However, I believe that this is not necessary due to the fact that the majority of website users have the psychological mind-set to scroll downwards once they have filled in input data such as the values of the component parameters.

## Navigation Map:





Whilst designing it, I decided to keep consistency. Consistency means that I had to make sure the user is able to navigate through the pages easily. The navigation bar at the top of the page is exactly the same on all pages within the website, this allows the user to easily know how to go to other pages within the site.

## Task Analysis:

(Task) The hierarchical analysis for persona one involves the user being able to *input the values into an ‘if statement’ component within the method section*:

Actions:

* Click and select an ‘if-statement’ block in the components section.
* Drag the unused component over to the method section.
* Click input values, and edit using the keyboard.

(Task) The hierarchical analysis for persona two involves the user being able to *use two components within the method section and then execute the method within the terminal*:

Actions:

* Click and select a component in the components section.
* Drag the unused component over to the method section.
* Click input values, and edit using the keyboard.
* Click and select a component in the components section.
* Drag the unused component over to the method section.
* Click input values, and edit using the keyboard.
* Scroll down.
* Click the ‘Run’ button.

After the task analysis, we can see that the probability of making a mistake is very low as most interactive components appear on demand after doing an action such as dragging a component from one section to another allowing the input values to appear and disappear where appropriate within the website.

## Design Decisions:

Most of the design decisions are explained beneath the sub-headings above. More on the consistency, I decided to use components such as panels throughout the website. The panels are used to highlight major points of the website functionality or important information to supplement things such as any other text within a page of the website.

In terms of colours, I kept using the same colours over and over again without making it look over-worked. Colours have been used from a created colour palette to keep consistency. The colour palette has different colours and different shades too.

Due to this being a prototype, I decided to keep the design only to be working on desktop computers with fairly large computers. This would be outlined in the client handover when giving access to the prototype.

In the mobile version, I would have utilised the Bootstrap CSS framework with its columns and rows to separate content on the screen when the user accesses the website on different screen sizes.

# 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 or further assistance is required.

# Usability Testing

## Test:

For the usability testing, I attended the practical lab session and got a peer to test the website functionality on a desktop environment.

The results and corresponding test results follow:

|  |  |
| --- | --- |
| **Test Case** | **[RESULT] Can the tester do it? (YES OR NO)** |
| Is the user able to navigate to the ‘About’ page with ease? | YES |
| Is the user able to drag and drop all the components from the Components section to the Method section? | YES |
| Is the user able to use one ‘if-statement’ and one ‘for-loop’ component and run the method in the terminal? | YES |
| Can the user close (the app), minimise and maximise the terminal? | YES |
| Does the user see a validation error (bottom right of screen) when they try to run a method with input values being empty? | YES |
| Can the user change a method parameter (an input value) and then re-run the program? | YES |

## Comments after test:

No comments necessary, user found it relatively easy to use the web application.

# Web Technology References and Site Access

To access the code repository, please use the following link:

<https://github.com/TingAli/University_CSCU9N5_Assignment_Design>

To access the website on a server, please use the following link:

<https://tingali.github.io/index.html>

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.