

Development Documentation

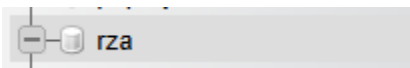
DANIELA DEICMANE

Development Documentation

The purpose of this document is to show different features in the software that have been developed and any changes made during the development process. It is important to note that most decisions made during the development were made to ensure the highest accessibility and success of the new software solution in improving the RZA reputation and increasing profits, market share and recognition.

Database and User Table


















Iteration 1



*Created database

```
CREATE TABLE users(  
    userID INT AUTO_INCREMENT PRIMARY KEY,  
    forename VARCHAR(100) NOT NULL,  
    surname VARCHAR(100) NOT NULL,  
    emailAddress VARCHAR(255) NOT NULL UNIQUE,  
    password VARCHAR(255) NOT NULL  
);
```

*Table to store user information (for log in)

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	userID 	int(11)			No	None		AUTO_INCREMENT	 Change  Drop  More
2	forename	varchar(100)	utf8mb4_general_ci		No	None			 Change  Drop  More
3	surname	varchar(100)	utf8mb4_general_ci		No	None			 Change  Drop  More
4	emailAddress 	varchar(255)	utf8mb4_general_ci		No	None			 Change  Drop  More
5	password	varchar(255)	utf8mb4_general_ci		No	None			 Change  Drop  More

*Users table

In this iteration we have created a new database named “rza” which stands for “Riget Zoo Adventures”, and we will store all the tables required for the RZA software within this database.

We have created a new table using MySQL within the XAMPP software named “users”, which will be used to store registered users data, including their forename, surname, email address and password. The “userID” has been set to integer data type and to “AUTO_INCREMENT” which means that it will be automatically generated when a new row

(user) has been added, also it is set to a “PRIMARY KEY” and so will be used to identify users. Furthermore, both the forename and surname have been set to “VARCHAR(100)” data type which means that it can include 100 characters including numbers, letters, and symbols, and they have also been set to “NOT NULL” which specifies that these fields cannot be empty. Additionally, the “emailAddress” has also been set to “VARCHAR(255)” but instead of 100 characters it enables a maximum of 255 characters, and it has also been set to “NOT NULL”, however it has also been specified that the email address must be “UNIQUE” which means that the email address must not match with any other email address within the table. Lastly, the password column has also been set to “VARCHAR(255)” and “NOT NULL”.

Iteration 2

```
<?php

class DatabaseConnection{

    //Get the values that are required to connect to the database
    private $server = "localhost";
    private $username = "root";
    private $dbPassword = "";
    private $dbName = "rza";
    //Variable to store the connection
    private $conn;

    //Makes connection
    public function connect(){
        //Sets the connection to null
        $this->conn = null;
        //Tries to make a connection
        try{
            $this->conn = new PDO("mysql:host=".$this->server. ";dbname=".$this->dbName, $this->username,
                                $this->dbPassword);
            $this->conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
        }
        //If the connection failed it will catch the error message and then display it
        catch(PDOException $e){
            echo "Connection error: ".$e->getMessage();
        }
        //return connection
        return $this->conn;
    }
}
```

*Class containing the database connection

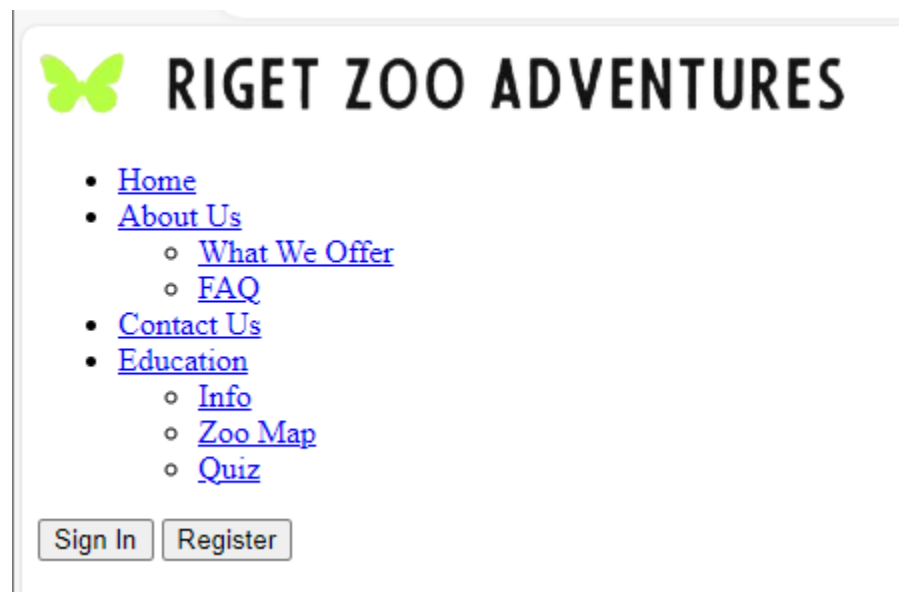
In this iteration we have made connection with the database that we previously created. To do so we used PHP object-oriented programming. We first created a “DatabaseConnection” class that will store the code until an object for that class is created and so will execute the code. Then we assigned the host, user, database password

and name to private variables which ensures that other classes cannot access these values and so increases the security of the data stored within the database. Then we created a public function called “connect()”, by setting it to public it ensures that the connection can be accessed, within this function it tries to make a connection using “PDO” if it is unsuccessful it will execute the “catch” statement which will get the error and display it. Irrespective of which statement is successfully executed it will either return the connection or the error message.

Navigation

In this section we will show how the navigation menu for RZA software is developed and designed. It is essential that it is visually appealing and functional as well as accessible as this is one of the first things that the user will see and interact with, therefore they might use it to judge the overall software.

Iteration 1



*Initial navigation menu

```

<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
</head>
<body>
  <!--Navigation-->
  <nav>
    <!--Logo-->
    <a href="index.php"></img></a>
    <!--Navigation menu-->
    <ul>
      <li><a href="#">Home</a></li>
      <li>
        <a href="#">About Us</a>
        <!--Dropdown menu-->
        <ul>
          <li><a href="#">What We Offer</a></li>
          <li><a href="#">FAQ</a></li>
        </ul>
      </li>
      <li><a href="#">Contact Us</a></li>
      <li>
        <a href="#">Education</a>
        <!--Dropdown Menu-->
        <ul>
          <li><a href="#">Info</a></li>
          <li><a href="#">Zoo Map</a></li>
          <li><a href="#">Quiz</a></li>
        </ul>
      </li>
    </ul>
    <!--Sign In and Register buttons-->
    <div>
      <button>Sign In</button>
      <button>Register</button>
    </div>
  </nav>
</body>
</html>

```

*Navigation Menu Code

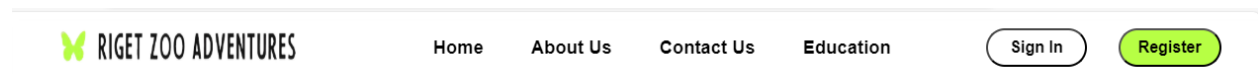
In this iteration we have just added the HTML file without any CSS styling. We have added the RZA company logo at the top and menu navigation options, where the “What We Offer” and “FAQ” are a sub-list for the “About Us” and the “Info”, “Zoo Map”, and “Quiz” are the

sub-list for the “Education” menu option. If the user clicked on any of these links it would show the homepage, as for now we have not developed any other feature and page.

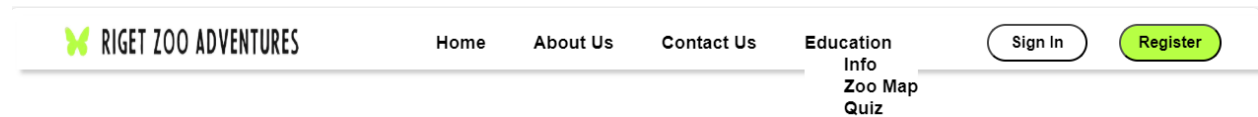
Also, we decided to use a butterfly for the symbol of the logo as it symbolise change and nature. Additionally, we colored it bright green (#B7FF44) to match with the rest of the design of the navigation menu and so form a branding.

When developing this section, we made sure to use descriptive tags, e.g., “img” and “ul”, as it will ensure a higher level of accessibility, especially for people using screen readers. Additionally, by using “button” tag instead of creating a custom button using “div” tag, it ensures that these buttons can be keyboard focused. Lastly, we also added “alt” text for the logo to ensure that when using assistive technology or screen readers it would read the alternative text, also for browsers that do not support this image type will show the text instead of the image.

Iteration 2



*Menu Navigation



*Menu Navigation with Activated Dropdown Menu

```

/*These styles will apply to all pages that include the styles file*/
*{
  margin: none;
  padding: none;
  background: white;
  font-family: sans-serif;
}

/*****Navigation*****/

/*Designs the navigation layout and add a shadow*/
#navigation{
  display: flex;
  position: fixed;
  width: 100vw;
  flex-direction: row;
  align-items: center;
  justify-content: space-around;
  background: transparent;
  box-shadow: 4px 4px 4px rgba(0, 0, 0, 0.2);
}

/*Removes the list style*/
#navigation ul{
  list-style-type: none;
}

/*Arranges home, about us, contact us, education menu items in row
and adds a gap between them*/
#main-nav{
  display: flex;
  flex-direction: row;
  gap: 3em;
}

/*Customises the navigation list items*/
.nav-links{
  /*Removes underline of the text*/
  text-decoration: none;
  color: black;
  font-weight: bold;
  /*Increases the font size of the text. 1 rem = default size (16px)*/
  font-size: 1.15rem;
}

/*Change the size of the logo*/
#nav-logo{
  height: 2.5rem;
  width: 15rem;
}

/*Customises how the dropdown navigation should be displayed*/
.dropdown{
  /*Prevents the dropdown menu from being displayed*/
  display: none;
  z-index: 999;
  /*Ensures that the dropdown menu is displayed without affecting the main menu*/
  position: absolute;
  flex-direction: column;
}

#main-nav li:hover ul.dropdown{
  /*Displays the menu if the menu item has been hovered over*/
  display: flex;
}

```

*CSS code for the navigation

```

/*Displays the sign in and register buttons in row and add a gapo between them*/
#nav-buttons{
    display: flex;
    flex-direction: row;
    gap: 2em;
}

/*Customises the sign in button in the navigation*/
#sign-in{
    /*Adds a gap between the text and the button outline*/
    padding: 6px 22px;
    background: transparent;
    filter: brightness(85%);
    border-radius: 30px;
    font-weight: bold;
    font-size: 1em;
}

/*Customises the register button in the navigation*/
#register{
    /*Adds a gap between the text and the button outline*/
    padding: 8px 18px;
    /*Adds a background color for the button*/
    background-color: #B7FF44;
    border-radius: 30px;
    font-weight: bold;
    font-size: 1em;
}

```

*CSS code for the navigation


```

<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">

  <!--Connects to the styles.css file-->
  <link rel="stylesheet" href="../css/style.css">
</head>
<body>
  <!--Navigation-->
  <nav id="navigation">
    <!--Logo-->
    <a href="index.php"></img></a>
    <!--Navigation menu-->
    <ul id="main-nav">
      <li><a class="nav-links" href="#">Home</a></li>
      <li>
        <a class="nav-links" href="#">About Us</a>
        <!--Dropdown menu-->
        <ul class="dropdown">
          <li><a class="nav-links" href="#">What We Offer</a></li>
          <li><a class="nav-links" href="#">FAQ</a></li>
        </ul>
      </li>
      <li><a class="nav-links" href="#">Contact Us</a></li>
      <li>
        <a class="nav-links" href="#">Education</a>
        <!--Dropdown Menu-->
        <ul class="dropdown">
          <li><a class="nav-links" href="#">Info</a></li>
          <li><a class="nav-links" href="#">Zoo Map</a></li>
          <li><a class="nav-links" href="#">Quiz</a></li>
        </ul>
      </li>
    </ul>
    <!--Sign In and Register buttons-->
    <div id="nav-buttons">
      <button id="sign-in">Sign In</button>
      <button id="register">Register</button>
    </div>
  </nav>
</body>
</html>

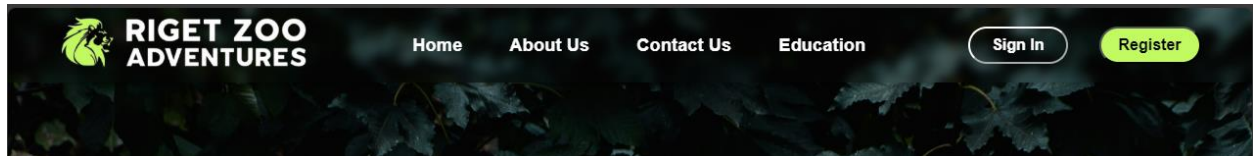
```

*HTML code for the navigation

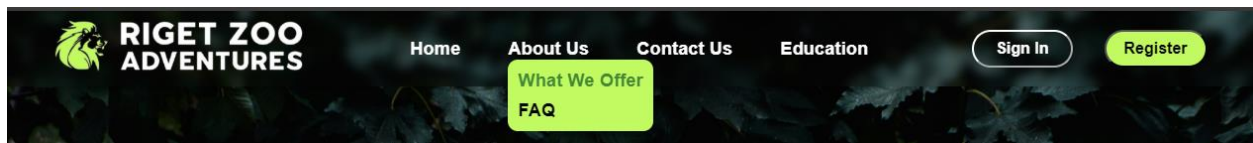
In this iteration we have added the layout of the navigation as well as the design of it, however the links will not perform an action still, but once the prototype is finished the links within the navigation will re-direct the user to the appropriate page.

We used a CSS flexbox to ensure that the navigation is responsive based on the screen size, and so the gap between the elements increases or decreases based on the screen size. Additionally, we added a shadow at the bottom of the navigation to enhance it and therefore make it stand out.

Iteration 3



*Main Navigation



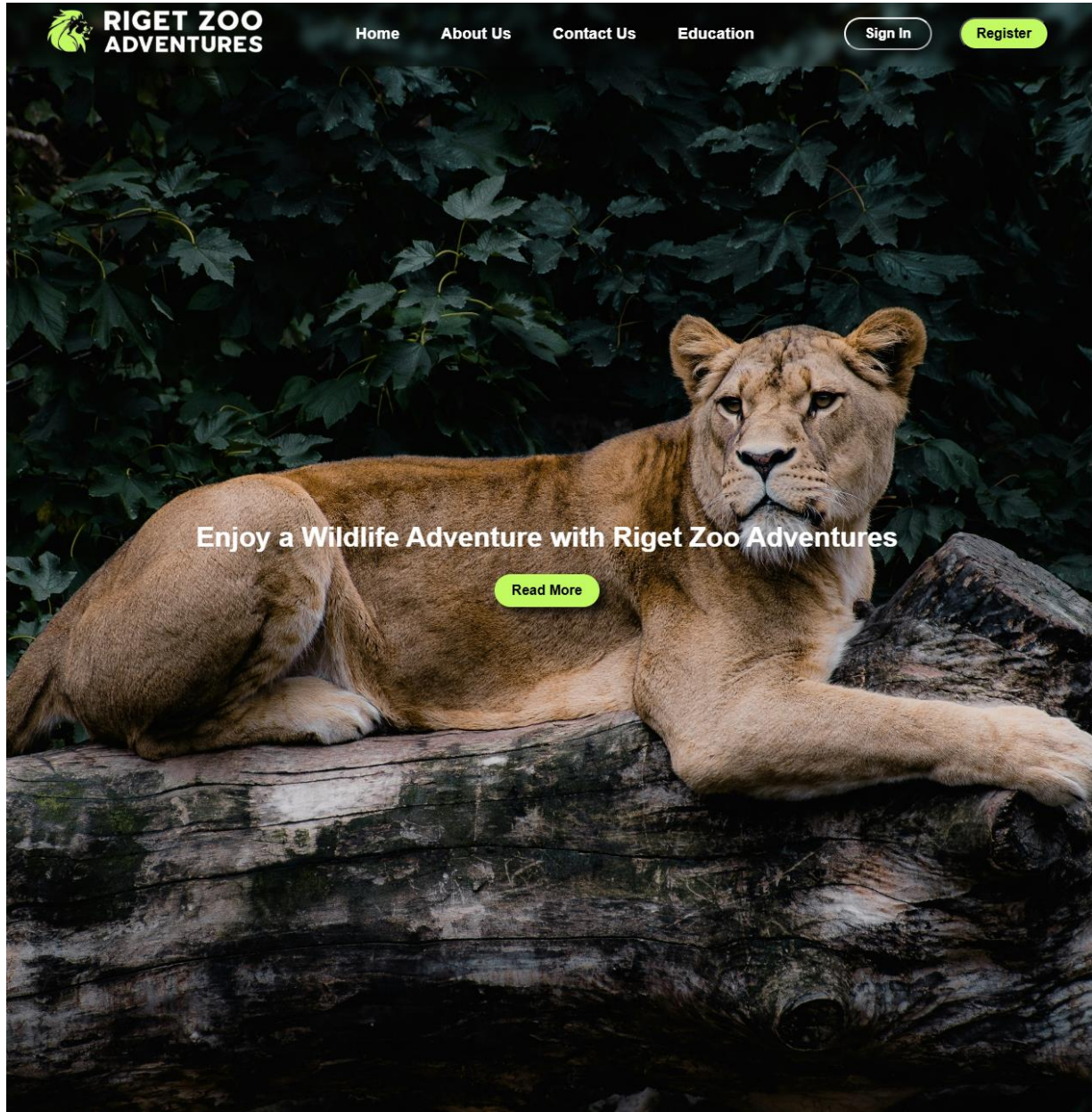
*Main navigation with activated dropdown menu

This is a final iteration for the navigation menu, as we have fully developed it. At this stage we decided to change the logo to a bit bigger text that is separated into two rows, this will enhance the logo and gain users attention, and a lion symbol instead of butterfly as lion represent wildlife. The lion symbol will still be in the bright green color to ensure that the software is consistent and so creates a better brand image. Additionally, we changed all text to white color as the background of the navigation is dark and blurred. Lastly, we decided to make the sub-menu color bright green so that it would not combine with the main menu, and the text within the sub-menu will be in black color instead of white to ensure the compliance with the WCAG requirements, but when the user hovers one of the options it will change to dark green color. It is important to note the main menu items will also change color, but to the bright green color, mainly due to the background color of the menu. This will ensure that the user is aware that clicking on any of the menu options will result in an action.

Homepage – Header

This is the first section of the homepage and so requires it to be appealing and make the user stay on the software for longer.

Iteration 1



*Homepage Header

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">

  <!--Connects to the styles.css file-->
  <link rel="stylesheet" href="../css/style.css">

  <title>Riget Zoo Adventures</title>
</head>
<body>

  <!--Included the navigation at the top of the page-->
  <?php
    include "navigation.php";
  ?>

  <!--Header-->
  <header class="homepage-image">
    <div>
      <!--Header text to attract the users-->
      <h1>Enjoy a Wildlife Adventure with Riget Zoo Adventures</h1>
      <!--CTA-->
      <button>Read More</button>
    </div>
  </header>

</body>
</html>
```

*HTML file for the homepage

```

/*****Homepage*****/

/*****Header*****/

.homepage-image{
  /*Added a background image to the header*/
  background-image: url('../static/mike-van-den-bos-gszvu4ZVWi0-unsplash.jpg');
  /*Prevent the image from repeating*/
  background-repeat: no-repeat;
  /*Center the image*/
  background-position: center;
  /*Cover the page*/
  background-size: cover;
  /*Fixes the image*/
  background-attachment: fixed;
  /*Ensures that the image covers all the width and
  all height based on the viewpoint*/
  width: 100%;
  height: 100vh;
}

/*Center the header text and CTA button*/
.homepage-image div{
  height: 100vh;
  display: flex;
  flex-direction: column;
  align-items: center;
  justify-content: center;
  /*Ensures that the text and button is on the top of the page*/
  z-index: 999;
  /*Gap between the text and the button*/
  gap: 1.5em;
}

/*Colors the header text in white color*/
.homepage-image div h1{
  color: white;
}

/*Customises the button*/
.homepage-image div button{
  /*Adds padding between the text and the border of the button*/
  padding: 10px 20px;
  /*Change the background color of the button*/
  background: #B7FF44;
  font-weight: bold;
  font-size: 1em;
  border-radius: 30px;
  /*Removes the border*/
  border: none;
  cursor: pointer;
  /*The first value is how much the shadow should be to the left side,
  the second how much it should be to the bottom, and the third is the blur*/
  filter: drop-shadow(8px 8px 4px rgba(0, 0, 0, 0.6));
}

```

*CSS for the homepage header

The header contains a single line text to clarify and contains the main message that emphasises the purpose of the software and the business. Furthermore, most users just skim through pages, especially the homepages, therefore having a short, single line text will ensure that the user can read it quickly and understand the core message. Lastly, it can also improve the business image, as it can be easier for the user to remember this message.

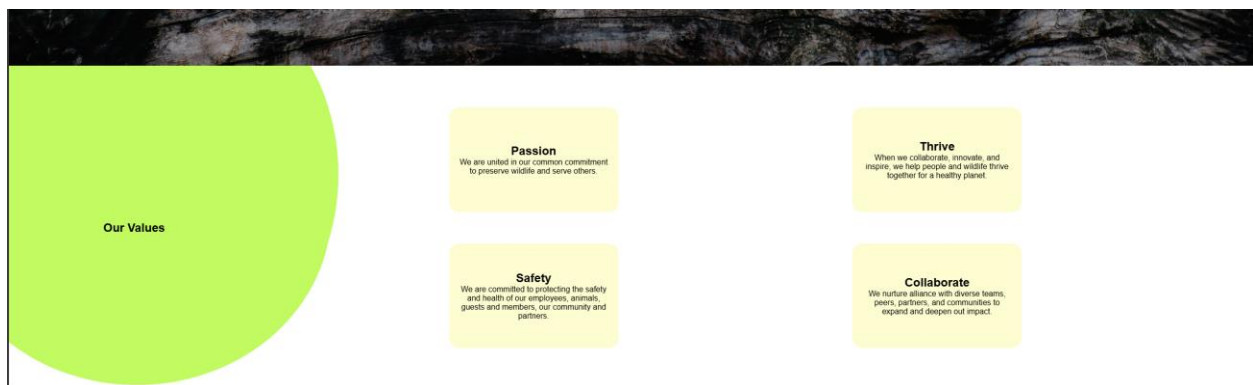
Additionally, we added the call to action (CTA) button to encourage the user to take immediate action, which in this case is to learn more about the business and what they

offer. This, therefore, can increase conversion rates (the transition from a user to a customer), as well as reduce the bounce rate (the rate at which the users view the software and exits before taking any action) which will ensure that the users will stay on the software for longer.

For the background image we choose a darker image as the navigation has a transparent blurred background, additionally the main color we are using in this website is bright green, which shows better on darker background.

Homepage – Values Section

Iteration 1



*Values section of the homepage


```

<!--The main body of the page-->
<main>
  <!--Values section-->
  <section id="background-image">
    <!--The content of the section-->
    <div id="values-section">
      <!--The header of the section-->
      <div id="values">
        <h1>Our Values</h1>
      </div>
      <!--The values-->
      <div id="values-list">
        <!--First pair of values-->
        <div id="first-list">
          <div class="value-item">
            <h2>Passion</h2>
            <p>We are united in our common commitment to preserve wildlife and serve o
          </div>
          <div class="value-item">
            <h2>Safety</h2>
            <p>We are committed to protecting the safety and health of our employees,
              guests and members, our community and partners.</p>
          </div>
        </div>
        <!--Second pair of values-->
        <div id="second-list">
          <div class="value-item">
            <h2>Thrive</h2>
            <p>When we collaborate, innovate, and inspire, we help people and wildlife
              for a healthy planet.</p>
          </div>
          <div class="value-item">
            <h2>Collaborate</h2>
            <p>We nurture alliance with diverse teams, peers, partners, and communitie
              and deepen out impact.</p>
          </div>
        </div>
      </div>
    </div>
  </section>
</main>

```

*HTML code for values section

```

/*****Values Section*****/

/*Adds the background image to the section and prevents it from repeating*/
#background-image{
    background-image: url('../static/values-image.png');
    background-repeat: no-repeat;
}

/*Overall section id used to display the two parts in row and
set the overall width to 100% of the page*/
#values-section{
    display: flex;
    flex-direction: row;
    /*Use % instead of viewpoint to prevent the scrollbar from being displayed*/
    width: 100%;
}

/*The heading of the section*/
#values{
    /*Places the heading to the center of the page and
    allocates 30% of the available width*/
    display: flex;
    align-items: center;
    /*Sets the height of the section*/
    height: 50vh;
    flex: 30%;
    /*Adds a padding to the left so that the header is not
    displayed at the start of the width*/
    padding-left: 12em;
}

/*Overall section for all 4 values*/
#values-list{
    /*Displays the pairs of values in row and the center of the section*/
    display: flex;
    flex-direction: row;
    justify-content: center;
    align-items: center;
    /*Allocates 70% of the available width*/
    flex: 70%;
}

/*Displays the both pairs in columns and allocates them with 35% of the width*/
#first-list, #second-list{
    display: flex;
    flex-direction: column;
    flex: 35%;
    /*Adds a gap between the two values in the pair*/
    gap: 4em;
}

```

*CSS code for the values section


```

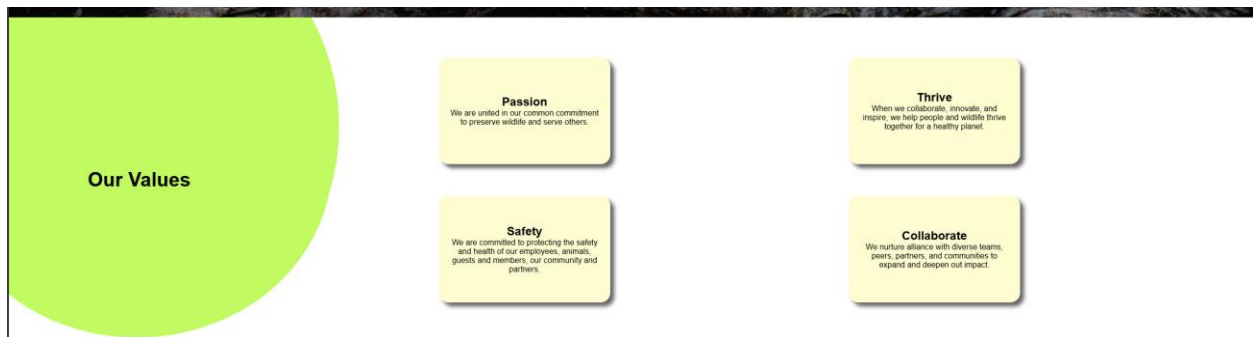
/*Customises the individual value sections*/
.value-item{
  /*Sets the width and height of the section*/
  width: 19em;
  height: 12em;
  /*Adds padding between the section border and the text, to ensure
  that there is no cluster*/
  padding: 10px 20px;
  /*Displays the text in the center of its section*/
  display: flex;
  flex-direction: column;
  justify-content: center;
  text-align: center;
  /*Sets the background color to light yellow*/
  background-color: #FFFDCE;
  /*Makes the border of the section rounded*/
  border-radius: 20px;
}

```

*Rest of the CSS code for the values section

In this section we have added a white background and a bright green circle on the left corner where the title will be. On the white background we have added four, equal sized boxes which contain information about the 4 values of RZA.

Iteration 2



*Updated "Values" section

```

<!--The header of the section-->
<div id="values">
  <h1 style="font-size: 2.5em;">Our Values</h1>
</div>

```

*Changed HTML code

```
/*The first value is how much the shadow should be to the left side,  
the second how much it should be to the bottom, and the third is the blur*/  
filter: drop-shadow(8px 8px 4px rgba(0, 0, 0, 0.6));
```

*Changed CSS code

In this iteration we increased the size of the heading so that it is more accessible and easier to read. Also, we added a shadow for each of the value containers to enhance them.

Homepage – Our Animals

Iteration 1

Our Animals



Cheetah

The cheetah is a large cat and the fastest land animal. It has tawny to creamy white or pale buff fur that is marked...



Brown Bear

Brown bears, also known as grizzly bears, are the largest terrestrial carnivores. They have a distinct shoulder hump, dish-shaped...



Green Tree Python

The green tree python, also known as the emerald green python, is a species of snake in the family Pythonidae. They have a diamond-shaped head, over 100 long, sharp, backward-pointing...

[Read More](#)

*Our animals section of the homepage

```

<!--Our Animals section-->
<section id="our-animals">
  <!--Section heading-->
  <div>
    <h1>Our Animals</h1>
  </div>
  <!--Main body of this section-->
  <div id="our-animals-info">
    <!--First image and text section-->
    <div class="our-animals-images-section">
      
      <!--Animal Name-->
      <h3>Cheetah</h3>
      <!--Info about the animal-->
      <p>The cheetah is a large cat and the fastest land animal. It has tawny to creamy white or pale buff fur that is marked...</p>
    </div>
    <!--Second image and text section-->
    <div class="our-animals-images-section">
      
      <!--Animal Name-->
      <h3>Brown Bear</h3>
      <!--Info about the animal-->
      <p>Brown bears, also known as grizzly bears, are the largest terrestrial carnivores. They have a distinct shoulder hump, dish-shaped...</p>
    </div>
    <!--Third image and text section-->
    <div class="our-animals-images-section">
      
      <!--Animal Name-->
      <h3>Green Tree Python</h3>
      <!--Info about the animal-->
      <p>The green tree python, also known as the emerald green python, is a species of snake in the family Pythonidae. They have a diamond-shaped head, over 100 long backward-pointing...</p>
    </div>
  </div>
  <!--CTA button-->
  <div>
    <a href="#"><button id="read-more-animals">Read More</button></a>
  </div>
</section>

```

*HTML code for “Our Animals” section

```

/*****Our Animals*****/

/*Display the content of our animals section in column and add a gap between each
element.*/
#our-animals{
  display: flex;
  flex-direction: column;
  margin: 5em;
  gap: 3em;
}

#read-more-animals{
  /*Adds padding between the text and the border of the button*/
  padding: 10px 20px;
  /*Change the background color of the button*/
  background: #B7FF44;
  font-weight: bold;
  font-size: 1em;
  border-radius: 30px;
  /*Removes the border*/
  border: none;
  cursor: pointer;
  /*The first value is how much the shadow should be to the left side,
  the second how much it should be to the bottom, and the third is the blur*/
  filter: drop-shadow(8px 8px 4px rgba(0, 0, 0, 0.6));
}

/*Displays the images and texts in row and adds a gap between each sub-section(image)*/
#our-animals-info{
  display: flex;
  flex-direction: row;
  justify-content: space-evenly;
  gap: 2em;
}

.our-animals-images{
  /*Specifies the size of the image using % to ensure that the images
  are responsive*/
  width: 80%;
  height: 70%;
  /*Adds a radius*/
  border-radius: 20px;
  /*The first value is how much the shadow should be to the left side,
  the second how much it should be to the bottom, and the third is the blur*/
  filter: drop-shadow(8px 8px 4px rgba(0, 0, 0, 0.6));
}

/*Displays each image and corresponding text in column and adds a gap between them*/
.our-animals-images-section{
  display: flex;
  flex-direction: column;
  gap: 1.5em;
}

```

*CSS code for “Our Animals” section

So far we have added a headline and images of three animals, Cheetah, Brown Bear, and Green Tree Python, and some information for each of them to make the users interested. We have also added the CTA button to encourage the user to read more about the animals that are within the RZA. By having some information for each animal and then adding three dots at the end it further encourages the user to click on the button.

Iteration 2

Our Animals



Cheetah

The cheetah is a large cat and the fastest land animal. It has tawny to creamy white or pale buff fur that is marked...

[Read More](#)



Brown Bear

Brown bears, also known as grizzly bears, are the largest terrestrial carnivores. They have a distinct shoulder hump, dish-shaped...



Green Tree Python

The green tree python, also known as the emerald green python, is a species of snake in the family Pythonidae. They have a diamond-shaped head, over 100 long, sharp, backward-pointing...

*Our Animals section

```
.our-animals-images{  
  /*Specifies the size of the image using % to ensure that the images  
  are responsive*/  
  width: 80%;  
  height: 75%;  
}
```

*Changes made to CSS code

```
/*Display the content of our animals section in column and add a gap between each  
element.*/  
#our-animals{  
  display: flex;  
  flex-direction: column;  
  margin: 5em;  
  gap: 2.5em;  
}
```

*Changes made to the CSS code

```
<!--Animal Name-->  
<h3 style="font-size: 1.7em;">Green Tree Python</h3>
```

*Changes made to the HTML code

In this iteration we increased the size of the heading to 2.5rem and the size of the animal names to 1.7em so that it would be more accessible and clearer. Additionally, we increased the height of the images so that they would look better and make them clearer. Lastly, we added a shadow for each of the images to make them look more like a 3D and enhance them.

Homepage – Our Prices and Footer

Iteration 1

Our Prices

Some Information regarding tickets



Single Ticket

Must be 16+
For one person
No time limit
Able to feed up to 2 animals

Select



Family Ticket

For two adults and one children
Children must be under 16
No time limit
Able to feed up to 4 animals

Select



Golden Membership

Must be 16+
Able to feed up to 7 animals
Unlimited visits
15% of all attractions

Select

This section will include information regarding some of the ticket types that the user can purchase, however if they have not logged in they will be brought to a new page which will ask the user to sign up or log in before they can purchase any of the tickets, otherwise they will be brought to a checkout. Each ticket type also includes an icon based on the ticket type, which improves the design and also makes the software more accessible as the users can use the icons to navigate around. Additionally, the “Select” buttons are in bright green color, and they have a shadow to make it stand out and therefore encourage the user to click on them and so purchase tickets. Lastly, we added a shadow around the middle option so that it creates a 3D design and improves the overall section design. Also, it ensures that each ticket type appears to be individual even without changing their background color.

The footer will include the “All rights reserved” which means that the information and the content within the software is protected and so cannot be used. Additionally, it will include links to “Terms and Conditions”, “Terms of Use”, “Private Policy”, which will ensure the compliance to the relevant legislation, including DPA as the user are aware of how their data will be used and stored, as well as how they are allowed to use the software, which will enable RZA to make the customer accountable if they do not comply with these requirements.

Contact Us

This page will enable the user to complete and send a form with any enquiry that they have, as well as see the RZA address, email address, phone number, and social media links as an alternative way of contacting them. This page is extremely important as it not only builds trust and relationship between the users and the company but also proves that the business is legitimate and so safe.

Iteration 1

RIGET ZOO ADVENTURES

Home About Us Contact Us Education Sign In Register

Get in touch

Visit us
Come to our store
Address

Send an email to us
Our friendly team is here to help
someone@example.com

Call us
Mon to Fri From 8am to 5pm
[999](tel:999)

Social media
Facebook Instagram Pinterest Twitter

Randomfirst Randomlast

random@gmail.com

1234567891


Tell us what we can help you with

Submit Form

*RZA Contact Us page

In this iteration, we have included the navigation by adding the PHP script, the main form, and additional information. On the left side of the page, it contains alternative ways that the user can contact the company, including their address, email address, phone number, and social media, whereas on the right side there is a form that the user can quickly and easily complete and submit for any queries. The main color used is still bright green to build brand image. However, for now we have added only a black colored background as the navigation has a transparent background, and so the menu would not be visible on white background, therefore because of this the labels for the form are not visible as they are in black color.

Iteration 2



HomeAbout UsContact UsEducation

Sign InRegister

Get in touch

Visit us

Come to our store

Address

Send an email to us

Our friendly team is here to help

someone@example.com

Call us

Mon to Fri From 9am to 5pm

020

Social media

[!\[\]\(ebfbf1b8c561eee84bef96b7cb561c97_img.jpg\)](#) [!\[\]\(50d373407f2c5c081345e8c6082a00ec_img.jpg\)](#) [!\[\]\(061b6502549143f4d9d6853594a01fd1_img.jpg\)](#) [!\[\]\(84f7baf1d1b9ce8325b8837f4b5243e2_img.jpg\)](#)

First Name

Last Name

Email

Phone Number

Message

Submit Form

*RZA contact us page


```

<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">

  <!--Download document for the icons-->
  <link rel="stylesheet"
href="https://unpkg.com/boxicons@latest/css/boxicons.min.css">

  <!--Connects to the styles.css file-->
  <link rel="stylesheet" href="../css/style.css">

  <!--Title of the page-->
  <title>Contact Us</title>
</head>
<body>
  <!--Includes the navigation-->
  <?php
  |   include "navigation.php";
  >

  <!--Container that will contain the form and information that the
user can use to contact the RZA-->
  <main id="container">
    <!--Adds a blurred background for the section-->
    <section id="container-background">
      <!--Contains the information that the user can use to contact the company-->
      <div id="left_container">
        <div id="backg">
          <!--Title-->
          <h1>Get in touch</h1>
          <!--Address-->
          <div>
            <h3 style="margin-bottom:3px;">Visit us</h3>
            <p>Come to our store</p>
            <p>Address</p>
          </div>

          <!--Email-->
          <div>
            <h3 style="margin-bottom:3px;">Send an email to us</h3>
            <p>Our friendly team is here to help</p>
            <p><a href="mailto:someone@example.com">someone@example.com</a></p>
          </div>

          <!--Phone number-->
          <div>
            <h3 style="margin-bottom:3px;">Call us</h3>
            <p>Mon to Fri From 8am to 5pm</p>
            <p><a href="tel:999">999</a></p>
          </div>
        </div>
      </div>
    </section>
  </main>

```

*HTML code for contact us page

```

</div>
<!--Social media icons/links-->
<div>
  <h3 style="margin-bottom:3px;">Social media</h3>
  <div id="icons">
    <i class='bx bxl-facebook-circle' ></i><i class='bx bxl-instagram' ></i>
    <i class='bx bxl-pinterest' ></i><i class='bx bxl-twitter' ></i>
  </div>
</div>
</div>
</div>
<!--Contact form-->
<div id="right_container">
  <form action="" method="post">
    <!--First and last name-->
    <div id="fullname">
      <div>
        <label for="forename">First Name</label><br>
        <input type="text" id="forename" name="forename" placeholder="Randomfirst" required>
      </div>
      <div>
        <label for="surname">Last Name</label><br>
        <input type="text" id="surname" name="surname" placeholder="Randomlast" required>
      </div>
    </div>
    <!--Email address-->
    <div>
      <label for="email">Email</label><br>
      <input type="email" id="email" name="email" placeholder="random@gmail.com" required>
    </div>
    <!--Phone number-->
    <div>
      <label for="phone">Phone Number</label><br>
      <input type="tel" id="phone" name="phone" placeholder="1234567891" required>
    </div>
    <!--Users message-->
    <div>
      <label for="message">Message</label><br>
      <textarea id="message" name="message" placeholder="Tell us what we can help you with" require></textarea>
    </div>
    <!--Submit button-->
    <button id="sbm_form" type="submit">Submit Form</button>
  </form>
</div>
</section>
</main>
</body>
</html>

```

*HTML code for contact us page

To improve the design and visibility we added a background image that is dark, but also lighter than just black color. Additionally, to make sure that the form is easy to read and is not blurred with the background we added a transparent, and blurred background to it, which will also enhance and make the main content stand out.

Additionally, we increased the size of the container left and so we had to position the form in the center to ensure the best design possible.

Registration Page

Iteration 1

Sign Up

Forename

Surname

Email Address

Password

Sign Up

Have an account: [Sign In](#)

[BACK TO HOME](#)

*Registration form

```

/*****Register*****/

/*Adds a background image for the registration page*/
#register-background{
  background-image: url("../static/bee-eater-9256610_1280.jpg");
  /*Prevents the image from repeating to cover the area*/
  background-repeat: no-repeat;
  /*Ensures that the image covers the whole area/page*/
  background-size: cover;
  /*Centers the image*/
  background-position: center;
  /*Fixes the image*/
  background-attachment: fixed;
}

/*Centers the form area and adds a gap between the "back to home" link and the form*/
#register-form-bg, #signin-form-bg{
  height: 100vh;
  display: flex;
  flex-direction: column;
  justify-content: center;
  align-items: center;
  gap: 3em;
}

/*Customises the registration form and sign in form*/
#register-form, #signin-form{
  /*Centers the form within its area and arranges the elements in the form in column and then adds
  a gap between each element*/
  display: flex;
  flex-direction: column;
  justify-content: center;
  align-items: center;
  gap: 2em;
  /*Adds padding between the border and the form*/
  padding: 20px 20px;
  /*Makes the background transparent*/
  background: transparent;
  /*Makes the background blurry*/
  backdrop-filter: blur(5px);
  /*Rounds the border of the form area*/
}

```

*HTML code for registration and sign in page

```

border-radius: 20px;
/*Adds a shadow around the form section/area and colors the border in black colour*/
box-shadow: 7px 7px 20px rgba(0, 0, 0, 0.1), -7px -7px 20px rgba(0, 0, 0, 0.1);
border: 1.5px solid black;
}

#sign-title{
  color: white;
  font-size: 2rem;
}

/*Arranges the input fields in columns and adds a gap between each of them*/
.sign-input-fields{
  display: flex;
  flex-direction: column;
  gap: 1em;
}

/*Arranges each of the label and input pairs in columns and adds
a gap between them*/
.sign-input-fields div{
  display: flex;
  flex-direction: column;
  gap: 0.5em;
}

/*Customise the label*/
.input-field label{
  font-size: 1.2rem;
  font-weight: bold;
  color: rgb(255, 255, 255);
}

/*Customise the input fields*/
.input-field input{
  outline: none;
  border: none;
  border-radius: 30px;
  background: rgba(209, 209, 209, 0.6);
  height: 2.3em;
  transition: 0.7s ease;
  padding: 3px 10px;
  width: 35em;
  font-weight: bold;
  font-size: 16px;
  cursor: pointer;
  color: white;
}

/*Adds a shadow around the curved input field/button
and changes background colour when hovered or focused*/
.input-field input:hover{
  background-color: #f4fee8;
  /*The first value is how much the shadow should be to the left side,
  the second how much it should be to the bottom, and the third is the blur*/
  filter: drop-shadow(8px 8px 4px rgb(65, 64, 64));
  color: black;
}

```

*HTML code for registration and sign in page

```

/*Adds a shadow, margin and changes background color when the user
focuses (clicks) on the input field*/
.input-field input:focus{
  background-color: ■ #f4fee8;
  /*The first value is how much the shadow should be to the left side,
  the second how much it should be to the bottom, and the third is the blur*/
  filter: drop-shadow(8px 8px 4px ■ rgb(65, 64, 64));
  margin-bottom: 1em;
  color: ■ black;
}

/*Customise the sign in and sign up buttons*/
.sign-up-button,
.sign-in-button{
  height: 2.5em;
  font-size: 16px;
  font-weight: bold;
  border-radius: 30px;
  border: none;
  outline: none;
  background-color: ■ #87FF44;
  color: ■ black;
  /*The first value is how much the shadow should be to the left side,
  the second how much it should be to the bottom, and the third is the blur*/
  filter: drop-shadow(8px 8px 4px ■ rgb(65, 64, 64));
  margin-top: 1.5em;
  cursor: pointer;
  transition: 1s ease;
}

/*Changes the background colour and text colour when the sign in or sign up button is hovered*/
.sign-up-button:hover,
.sign-in-button:hover{
  background-color: ■ #f4fee8;
  color: ■ black;
}

/*Customises the text*/
.have-an-account,
.dont-have-an-account{
  font-weight: bold;
  color: ■ white;
  font-size: 1.1rem;
}

/*Customises the link text*/
.sign-link-redirect{
  text-decoration: none;
  color: ■ rgb(155, 170, 255);
}

/*Customise error messages*/
.error-message{
  color: ■ red;
  font-weight: bold;
}

```

*HTML code for registration and sign in page

```

.back-to-home{
  /*Sets the color for the "BACK TO HOME" link*/
  color: ■rgb(255, 255, 255);
  /*Make the link text bold*/
  font-weight: bold;
  /*Changes the size of the link text*/
  font-size: 19px;
}

/*****Sign In*****/

/*Add a background image for the sign in page*/
#signin-background{
  background-image: url("../static/tiger-6047141_1280.jpg");
  /*Prevents the image from repeating to cover the area*/
  background-repeat: no-repeat;
  /*Adjusts the size of the background image to cover the area*/
  background-size: cover;
  /*Centers the image*/
  background-position: center;
  /*Fixes the image so that it wouldn't move*/
  background-attachment: fixed;
}

```

*HTML code for registration and sign in page

For the registration and sign in page we have added the company logo on the top left corner which also works as a link, since by clicking on it the user will be re-directed to the homepage, however to make the website more accessible we also added a link with “BACK TO HOME” text, if the user is not aware that the logo can re-direct them to the homepage. This also makes the software more accessible and easier to navigate around.

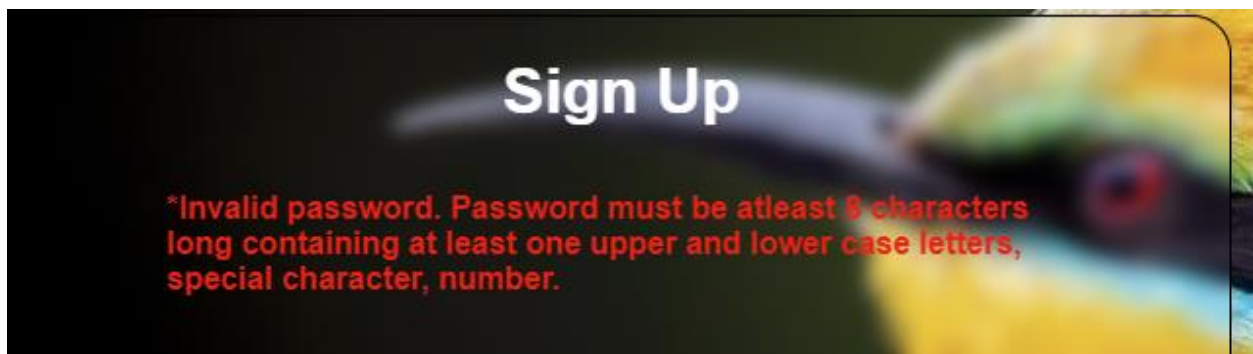
Furthermore, the sign-up form requires the user to enter their name, last name, email address, and their password before they can sign up, whilst the sign in form requires the user to enter their email address and password that is linked with their account. By clicking on any of the input fields it will bring the input field higher up and change the background color to a very light green color. Also, if the user hovers over the submit button it will also change background color to light green. Once the user has submitted either of the forms and action will be performed to either sign the user to the database by adding their information to the database table or by logging them into their account. Also, before any action can occur all the input will be tested, and an appropriate error message will be displayed to the user.

Additionally, we decided to color the “sign in” and “register” links in light blue color as dark blue color would not meet the required contrast level and so reduce the software

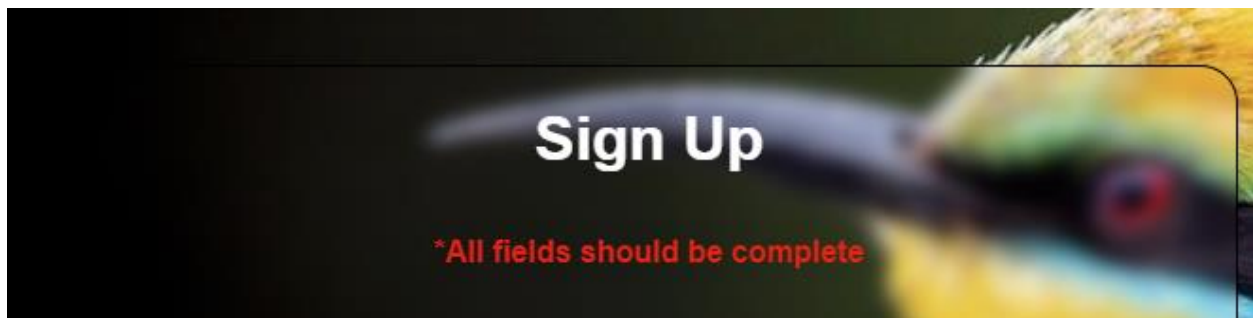
accessibility and visibility of the text. Once the user clicks on the “sign in” link they will be re-directed to the sign in page, whereas if the user clicks on the “register” link they will be re-directed to the registration page, which makes it easier to navigate around the software, and therefore improves user experience and promotes accessibility.

Lastly, to make the registration and sign in form stand out from the background we added a black border around the form container and the background color of the container is transparent and blurred, so that it would not combine with the background image, which could again affect the visibility of the form labels.

Iteration 2



*Example of error message



*Example of error message

As each input field is validated it will display an error message only for the first error they have identified. If the user fixes that error and then submits the form again with an invalid input in different input field that was also present previously, it will be displayed only now, since the user has solved the previous error.


```

<?php
    //Checks if the error has been set (If the error message is present)
    if(isset($_GET['error'])){
?>
        <!--Cleans the error message for security, especially against sql injections-->
        <p class="error-message">
            <?= $errors_present = UserValidation::clean($_GET['error']);?>
        </p>
    }
?>

```

*PHP code in register.php file

This code is inserted within the HTML code for the registration page and is used to check if any errors are present, and if it true then it will clean these errors to increase the security of the data, especially in case of SQL injections. Then this error will be displayed on red color to differentiate against other content, also red is associated with something negative, therefore will encourage the user to fix these errors.

```

<?php

//Gets the files for the database connection, validation file, and database sql queries
include "../core/connection.php";
include "../models/signup-validation.php";
include "../controller/signup-contr.php";

//Checks if the user has submitted the registration form
if(isset($_POST["sign-up"])){

    //Takes the data from the registration form
    //Carry out simple validation
    //The "Validation" is the class name whilst the "clean" is the static function
    //name. To carry out this the function
    $forename = UserValidation::clean($_POST["forename"]);
    $surname = UserValidation::clean($_POST["surname"]);
    $email = UserValidation::clean($_POST["email-address"]);
    $password = UserValidation::clean($_POST["password"]);

    //Checks if any of the fields are left empty, if it returns true
    //it will display an error message to indicate that the user hasn't filled all fields
    if(empty($forename) || empty($surname) || empty($email) || empty($password)){
        $em = "All fields should be complete";
        header("Location: ../view/register.php?error=$em");
    }

    //If the previous statement is not executed then it will check if the forename does
    //not consist of only letters. If it returns true then it will display a usefull
    //message indicating the fault to the user
    else if(!UserValidation::name($forename)){
        $forenameError = "Invalid Forename. Forename must consists of at least 2 characters
        and contain only letters.";
        header("Location: ../view/register.php?error=$forenameError");
    }

    //If the previous statement is not executed then it will check if the surname does
    //not consist of only letters. If it returns true then it will display a usefull
    //message indicating the fault to the user
    else if(!UserValidation::name($surname)){
        $surnameError = "Invalid surname. Surname must consists of at least 2 characters
        and contain only letters.";
        header("Location: ../view/register.php?error=$surnameError");
    }

    //If the previous statement is not executed then it will check if the email
    //is not in correct format. If it returns true then it will display a usefull
    //message indicating the fault to the user
    else if(!UserValidation::email($email)){
        $emailError = "Invalid email. Email must be in correct format.";
        header("Location: ../view/register.php?error=$emailError");
    }

    //If the previous statement is not executed then it will check if the password
    //is not in correct format. If it returns true then it will display a usefull
    //message indicating the fault to the user
    else if(!UserValidation::password($password)){
        $passwordError = "Invalid password. Password must be atleast 8 characters long containing
        at least one upper and lower case letters, special character, number.";
        header("Location: ../view/register.php?error=$passwordError");
    }
}

```

*PHP code to check any present errors

```

//If all the previous statements are not executed then it will check if the email
//already exist within the database. If it returns true then it will display a usefull
//message indicating the fault to the user
else{
    //Creates a new database execution
    $db = new DatabaseConnection();
    $conn = $db->connect();
    if($conn){
        //Creates a new object of the class and gives the connection to the class
        $emailExist = new ExistingEmail($conn);
        if(ExistingEmail::existingEmail($email)){
            $existingEmailError = "*Email has already been registered";
            header("Location: ../view/register.php?error=$existingEmailError");
        }
        //If the email doesn't exist within the database it will then send the
        //data to the database table to be stored
        else{
            //Creates a new object of the class and send the database connection to the class
            $user = new SignContr($conn);
            //Hashed password
            $password = password_hash($password, PASSWORD_DEFAULT);
            //Stores the data that needs to be inserted within the database in a array
            $data = [$forename, $surname, $email, $password];
            //Executes the function within the class and send the above array to it
            $res = $user->insert($data);
            //Checks if the data was stored successfully, if it returns true it will
            //redirect the user to the homepage
            if($res){
                header("Location: ../view/signin.php");
            }
            //If the data couldn't be inserted within the database it will display an error to the user
            //to make them aware of it (improved UI and UX)
            else{
                $em = "An error occurred!";
                header("Location: ../view/register.php?error=$em");
            }
        }
    }
}
}
}
?>

```

*PHP code to check for any errors

This file will clean all the data that has been received from the form and then clean it to ensure that it does not include any malicious code, e.g., to execute SQL injection. Once the data has been cleaned it will check whether any of the fields have been left empty. If it returns true it will send the error message to the registration page, but only once the user has submitted the form will the error message be displayed. However, if the user returns false it will validate the other fields consecutively, if no “else if” statement has returned true then it will check if the email already exists within the database if it returns false it will hash the password and store it within the database whilst redirecting the user to the log in page, however if it does return true then it will display the error message to indicate so.

Additionally, for better user experience and more robust software, we have added an “else” statement to check if any error has occurred during this process which then will be displayed to the user instead of crashing the whole software.

```
<?php
class UserValidation{

    //Cleans the data received from the user to increase security of sql injections
    static function clean($str){
        //Removes whitespaces or other pre-defined characters form
        //both sides of the data
        $str = trim($str);
        //Returns data with backslashes taken off "/"
        $str = stripslashes($str);
        //Converts special caharcters to HTML entities
        $str = htmlspecialchars($str);
        //Returns the string
        return $str;
    }

    //Validates forename and surname
    static function name($str){
        //Checks if the name consists of only letters
        $pattern = "/^[a-zA-Z]+$/";
        if(preg_match($pattern, $str)){
            //Checks if the name is at least 2 characters long
            if(strlen($str) >= 2){
                return true;
            }
        }
        else{
            return false;
        }
    }

    //Validates email format
    static function email($str){
        //Checks if the email is in correct format (conatins "@" and ".")
        if(filter_var($str, FILTER_VALIDATE_EMAIL)){
            return true;
        }
        else{
            return false;
        }
    }

    //Validates Password
    static function password($str){
        //Checks if the password consists of at least 8 characters from which at least
        //one is a number, a speacial character, upper case letter, and a lower case letter
        $pattern = '/^(?=.*[a-z])(?=.*[A-Z])(?=.*\d)(?=.*[\W_]).{8,}$/';
        if(preg_match($pattern, $str)){
            return true;
        }
        else{
            return false;
        }
    }
}

?>
```

*Sign Up validation

This class contains four different functions that are used to validate and clean the data sent from the registration page. The first function is used to remove whitespaces or other pre-defined characters, and it takes off the backslashes, and lastly it converts the special

characters to HTML entities, this ensures higher level of security and prevents cyber-attacks like SQL injections.

The second function is used to check if the forename and the surname contains only letters, either upper or lower case and that it is at least 2 characters long.

The third function checks if the email is in correct format, this means that it should have some text then “@” and then again text and then a “.” and again text.

Lastly, the fourth function checks if the password consists of at least 8 characters, from which at least one is upper- and lower-case letter, special symbol and a number.

```
<?php

class SignupContr{
    //Set variables to store the connection and table name
    private $table_name;
    private $conn;

    //Set variables to store all the values from the table
    private $userID;
    private $forename;
    private $surname;
    private $emailAddress;
    private $password;

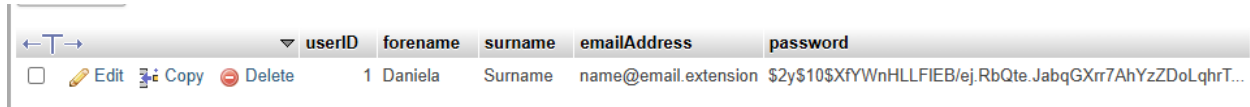
    //“__construct” function initialise the object's properties upon creating the object
    function __construct($db_conn){
        //Tkes the variable created above and stores the database connection
        $this->conn = $db_conn;
        //Stores the database table name from the variable above
        $this->table_name = "users";
    }

    //Function to add user data to the table
    function insert($data){
        //It will first try to insert the data and then if it doesn't work it will get
        //the error and return 0, which means that the error should not be displayed
        try{
            //Establishes the SQL statement that will need to be carried out
            //" $this->table_name" will take the above established table name
            //The question marks are used instead of the data variables for increased security
            //especially agains sql injections
            $sql = 'INSERT INTO ' . $this->table_name . '(forename, surname, emailAddress, password) VALUES(?, ?, ?, ?)';
            //Takes the established connection and prepares the above statement
            $stmt = $this->conn->prepare($sql);
            $res = $stmt->execute($data);
            return $res;
        }
        catch(PDOException $e){
            return 0;
        }
    }
}
```

*PHP code to receive database connection and insert data into database

This class first sets private variables to store sensitive information, including the user data and the connection to the database, and the table within the database to ensure a higher level of security. Then the “__construct()” function receives the database connection

variable from the “registration_errors.php” file and stores into previously set “\$conn” private variable and assigns the table name to the previously set “\$table_name” private variable.



The image shows a screenshot of a database management interface. At the top, there is a header bar with a dropdown menu and several column names: userID, forename, surname, emailAddress, and password. Below the header, there is a table with one row of data. The first column (userID) contains the value '1'. The second column (forename) contains 'Daniela'. The third column (surname) contains 'Surname'. The fourth column (emailAddress) contains 'name@email.extension'. The fifth column (password) contains a long, complex string: '\$2y\$10\$XYWnHLLFIEB/ej.RbQte.JabqGXrr7AhYzZDoLqhrT...'. To the left of the table, there are icons for 'Edit', 'Copy', and 'Delete'.

	userID	forename	surname	emailAddress	password
<input type="checkbox"/> Edit Copy Delete	1	Daniela	Surname	name@email.extension	\$2y\$10\$XYWnHLLFIEB/ej.RbQte.JabqGXrr7AhYzZDoLqhrT...

*Example of inserted data

The “insert()” function receives the user data from the “registration-errors.php” file and inserts it within the database, if an error occurs and the data could not be inserted the error will be returned.

```

class ExistingEmail{

    //Creates an empty variable to store database connection
    //To access the variable latter a static variable needs to be created
    private static $conn;

    // "__construct" function initilise the object's properties upon creating the object
    function __construct($db_conn){
        //Tkes the variable created above and stores the database connection
        //Use "self::" to access the static variable
        self::$conn = $db_conn;
    }

    //Checks if the email is already registered
    static function existingEmail($str){
        //Creates an SQL statement
        //It uses "?" for increased security, especially against SQL injections
        $sql = "SELECT COUNT(*) FROM users WHERE emailAddress = ?";
        //Uses "self::" to access the variable used to store connection and
        //prepares the statement for execution
        $stmt = self::$conn->prepare($sql);
        //Executes the statement and inserts the email in the position of the "?"
        $stmt->execute([$str]);
        //Fetches the returned columns
        $rows = $stmt->fetchColumn();

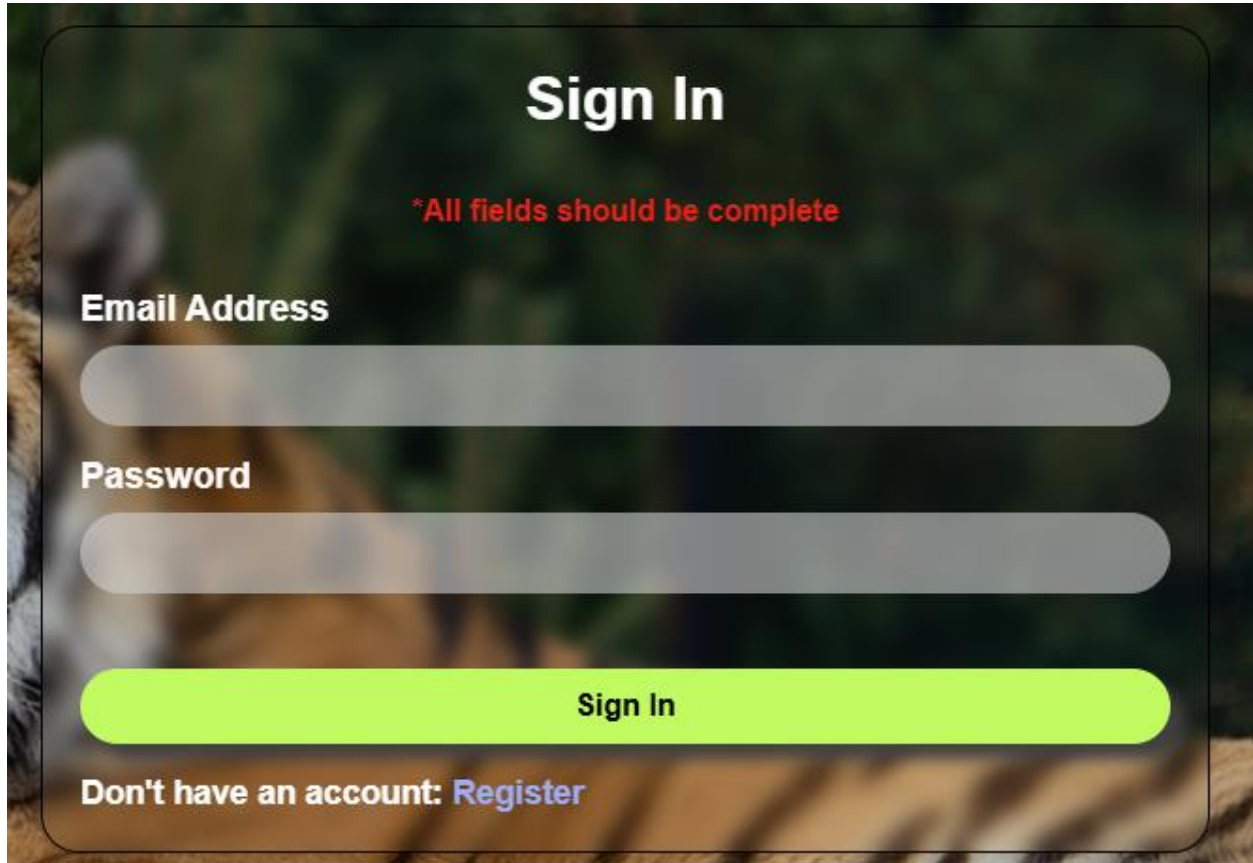
        //Checks if the number of rows returned is over 0 (which means that the email has
        //already been registered) and then it will return true, otherwise it will return false
        if($rows > 0){
            return true;
        }
        else{
            return false;
        }
    }
}

```

*PHP code to check if the email exists within the database

This section of code first receives the database connection and then executes the SQL statement to count how many occurrences of the specified email address are there within the “users” table in “emailAddress” column and then this statement is prepared with the connection and executed. Then the results are fetched into columns, so that the program could check how many occurrences were there since this PHP build in function returns just a single specified column from the table. Then using “if” statement it check if there are more than 0 rows returned, if it returns true it will also return true and so an error will be displayed to the user indicating that the email already exists, whilst if it returns false then the “else” statement will also return false and so an error would not be displayed.

Iteration 3



Sign In

***All fields should be complete**

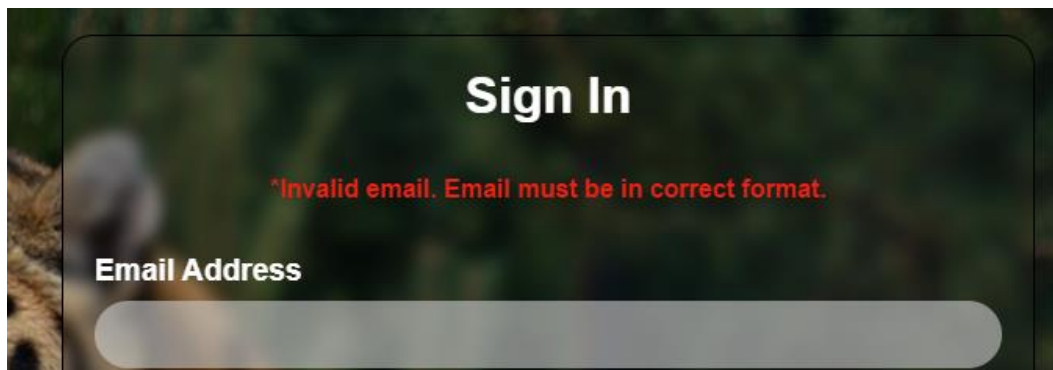
Email Address

Password

Sign In

Don't have an account: [Register](#)

*Sign In page error message



Sign In

***Invalid email. Email must be in correct format.**

Email Address

*Sign In error message


```

<?php
    //Start session so that data can be stored and retrieved from
    //$_SESSION variable
    session_start();

    //Gets the files for the database connection, validation file, and database sql queries
    include "../core/connection.php";
    include "../models/signup-validation.php";
    include "../controller/sign-contr.php";

    //Checks if the user has submitted the registration form
    if(isset($_POST["sign-in"])){

        //Takes the data from the registration form
        //Carry out simple validation
        //The "UserValidation" is the class name whilst the "clean" is the static function
        //name. To carry out this the function
        $email = UserValidation::clean($_POST["email-address"]);
        $password = UserValidation::clean($_POST["password"]);

        //Checks if any of the fields are left empty, if it returns true
        //it will display an error message to indicate that the user hasn't filled all fields
        if(empty($email) || empty($password)){
            $em = "**All fields should be complete";
            header("Location: ../view/signin.php?error=$em");
        }
        //If the previous statement is not executed then it will check if the email
        //is not in correct format. If it returns true then it will display a usefull
        //message indicating the fault to the user
        else if(!UserValidation::email($email)){
            $emailError = "**Invalid email. Email must be in correct format.";
            header("Location: ../view/signin.php?error=$emailError");
        }
        //If the previous statement is not executed then it will check if the password
        //is not in correct format. If it returns true then it will display a usefull
        //message indicating the fault to the user
        else if(!UserValidation::password($password)){
            $passwordError = "**Invalid password. Password must be atleast 8 characters long containing
            header("Location: ../view/signin.php?error=$passwordError");
        }
        else{

```

*PHP code to store errors

```

    else{
        //Creates a new database execution
        $db = new DatabaseConnection();
        $conn = $db->connect();
        //Creates a new object of a class and send the database connection to it
        $user = new SignContr($conn);
        //Executes the authentication function
        $auth = $user->auth($email, $password);
        //Check if the authentication was successful
        if($auth){
            //Stores user data within a variable
            $data = $user->getUser();
            //Assigns user email and id to session (used to identify user for a session)
            $_SESSION["id"] = $data["userID"];
            $_SESSION["email"] = $data["emailAddress"];
            //Redirects user to the account page
            header("Location: ../view/account.php");
        }
        else{
            //Returns an error if the email or password is incorrect
            $em = "*Incorrect email or password";
            header("Location: ../view/signin.php?error=$em");
        }
    }
}
?>

```

*PHP code to check credentials

This code starts session so that values can be stored within the PHP global “\$_SESSION” variable, then it includes all the files that will be required within this file.

The “if” statement checks if the user has submitted the form, if it returns true the data will be cleaned and then validated to ensure that no fields are left empty or if they are not in correct format. If all of these “if” and “else if” statements return false the “else” statement will be executed which creates a new database connection instance and sends this connection to the “SignContr” class, then using this connection the email and password are sent to the “auth()” function to check if the email and password are in the database and if they match the.

Then the nested “if” statement will be executed to check if it matches and are within the database, if it returns true then it will take the data within the array from “getUser” function and store the email and id into session variables and then the user will be re-directed to the account page. However, if it returns false an error message will be displayed to indicate that email or the password is incorrect.

```

//Authenticates log in email and password
function auth($emailAddress, $password){
    //Tries to execute sql statement
    try{
        //SQL statement that will need to be executed
        //Used "?" for higher security against SQL injections
        $sql = "SELECT * FROM " . $this->table_name . " WHERE emailAddress = ?";
        //Prepares the statements and connection
        $stmt = $this->conn->prepare($sql);
        //Replaces the "?" with the email address
        $res = $stmt->execute([$emailAddress]);
        //Checks if at least one row is returned
        if($stmt->rowCount() == 1){
            //Returns the results as associated array
            $user = $stmt->fetch();
            //Assigns the extracted data values to variables
            $db_email = $user["emailAddress"];
            $db_password = $user["password"];
            $db_id = $user["userID"];
            $db_forename = $user["forename"];
            $db_surname = $user["surname"];
            //Checks if the user entered email matches the database extracted email
            if($db_email === $emailAddress){
                //Checks if the user entered password in log in form
                //is the same as the hashed password stored in the table
                if(password_verify($password, $db_password)){
                    //Assigns the database extracted data to the above created variables
                    $this->emailAddress = $db_email;
                    $this->password = $db_password;
                    $this->userID = $db_id;
                    $this->forename = $db_forename;
                    $this->surname = $db_surname;
                    //Returns 1 for true
                    return 1;
                }
                //Otherwise it will return 0
                else{
                    return 0;
                }
            }
            //Otherwise it will return 0
            else{
                return 0;
            }
        }
        //Otherwise it will return 0
        else{
            return 0;
        }
    }
    //If the sql statement couldn't be executed it will return 0
    catch(PDOException $e){
        return 0;
    }
}

```

*PHP code for authentication

The “try” statement first prepares SQL statement using “?” instead of the exact value as this increases the security, then using the built-in function “execute” the “?” is replaced with the actual value and then the statement is executed. Then the nested “if” statement checks if there is just one row returned, if it returns true it will fetch the results and then assign these values to new variables which will be used to check if the email in this row matches the user entered email, if this “if” statement also returns true then next “if” statement will be executed to check if the stored password matches user entered password by using the PHP built-in function “password_verify”, as the password has been

hashed before it was stored in the database and it cannot be reversed back to original state. If this statement also returns true, the values within the row will be assigned to the initially set private variable. However, if it returns false, a value of “0” will be returned to the “signin-error.php” file and so an error message will be displayed, same will happen if the “try” statement fails.

```
//Places the information/data within an array
function getUser(){
    $data = array("userID" => $this->userID,
                  "forename" => $this->forename,
                  "surname" => $this->surname,
                  "emailAddress" => $this->emailAddress);
    return $data;
}
```

*PHP code to store user data within an array

This function assigns the user data to “key” and “pair” values and then returns this array. This array then is used in the “signin-error.php” file to assign the email and id to the session variables.

Account Page

Iteration 1

Personal Information

First Name*

Last Name*

log out

Account Information

Email Address*

[Change Password](#)

*Account Page

```

<body>

<?php
    //Creates an object/instance of database connection
    $db = new DatabaseConnection();

    //Gets the user data for displaying
    //Creates a new object for the class and sends a dataconnection to it
    $data = new SignContr($db->connect());
    //Executes the "getUserData" function and send the id
    $dataList = $data->getUserData($_SESSION["id"]);
?>

<!--Main body of the account page(profile information)-->
<main class="profile">
    <!--Displays the personal information about the registered user-->
    <section id="personal_info_container">
        <div id="personal_info">
            <h1>Personal Information</h1>
            <!--Displays the forename associated with the account-->
            <div>
                <label for="customerName">First Name<span style="color:red">*</span></label><br>
                <!--Using "=" instead of "php" is the same as saying "php echo"-->
                <input type="text" name="firstname" id="customerName" value="<?=$dataList["forename"]; ?>" readonly>
            </div>
            <!--Displays the surname associated with the account-->
            <div>
                <label for="customerLast">Last Name<span style="color:red">*</span></label><br>
                <input type="text" name="lastname" id="customerLast" value="<?=$dataList["surname"]; ?>" readonly>
            </div>
            </div>
            <!--Log out button-->
            <div id="logout_container">
                <form action="../models/log-out-user.php" method="post">
                    <button id="logout" type="submit" name="log-out">log out</button>
                </form>
            </div>
        </section>
        <!--Second section/part of the account/profile page-->
        <section id="account_info">
            <!--Information regarding the account-->
            <h1>Account Information</h1>
            <!--Displays the email address associated with the account-->
            <div>
                <label for="email">Email Address<span style="color:red">*</span></label><br>
                <input type="email" name="email" id="email" value="<?=$dataList["emailAddress"]; ?>" readonly>
            </div>
            <!--Link to change password-->
            <div>
                <a id="password" href="#" target="_self">Change Password</a>
            </div>
        </section>
    </main>
</body>

```

*HTML code for account page

The account page has two separate sections; the personal information section which displays information about the user, and the account information section which displays information that is associated with the account. Then at the bottom there is a button for the user to log out, which has a gradient background from bright green to light green so that it would stand out and so improve the user experience.