Statement of Purpose

Computers have many parts, and everyone uses their computer for different reasons. It's time-consuming having to browse through hundreds of different computers on multiple websites just to find the right one. Let alone the right one for the right price. You can never find a computer that has everything you need without making some sort of compromise. A computer with everything you need is critical to doing your job and/or schoolwork efficiently.

Using an innovative website with web scraping, we can provide custom computers at an amazing price without any compromises. With this website, we can allow customers to save both time and money while also ensuring they get everything they need. Due to the high demand for custom computers, an innovative way to customize these computers is crucial to cutting costs for consumers, saving time, and eliminating any compromises.

Research and Background

As someone who builds and repairs computers as a side gig. It became a must for me to have my own website where I could serve my customers’ needs quickly and efficiently. That is how I came up with this project. There are many websites that allow you to buy a pre-built PC, but most of the parts are not interchangeable. Even if they are, the price on them is locked because that company is keeping inventory of them. By creating a website that web scrapes the prices, I can make sure my customers are getting the most up to date deals.

The research I did to complete this project consisted of watching multiple YouTube videos about using both Beautiful Soup and Laravel. I started by following basics tutorials online and then implementing what I learned into my own project.

Project Languages, Software, and Hardware

* HTML
* JavaScript
* PHP
* MySQL
* CSS
* Python
* Laravel

Project Requirements

A screenshot of a computer program

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

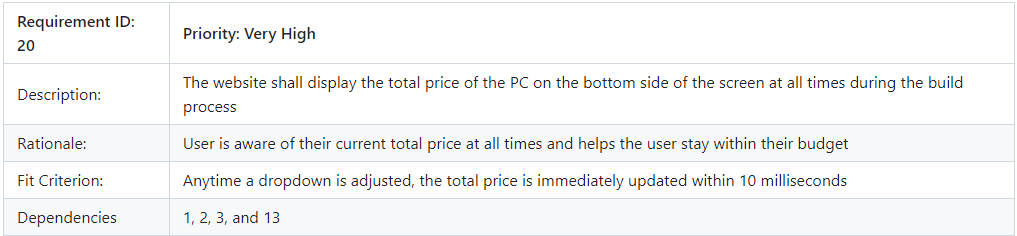
Description automatically generated

A screenshot of a survey

Description automatically generated

A screenshot of a computer

Description automatically generated



Project Implementation Description and Explanation

A screen shot of a computer

Description automatically generated <https://github.com/Natemixon7/CSU-Senior-Project/tree/master/src>

Fig. 1 Welcome Page

This is the welcome page users first see when they visit the website. This page consists of a login and register button in the top right corner. This is to direct users to easily login or register. The center of the page uses an eye-catching photo with a button in the center of it for users to get straight to building their PC.

A screen shot of a computer

Description automatically generated

Fig. 2 Login Page

A screenshot of a computer

Description automatically generatedThis login screen is sleek and simple to allow users a quick login process. Everything is rounded to provide a crisp design and error messages are provided to help guide users through their login process. The “Remember Me” box is optional so that users do not have to log in every time they visit the site. The “Forgot Your Password” allows users to reset their password should they not remember it. It also highlights when the user hovers over it, so they know it is clickable. A register button is provided in the bottom left corner so that if a user needs to register, they can quickly redirect to the register page. Error messages are in place for password do not match or email is not recognized. The links in the top right corner of the web page are still provided to users so they can easily navigate throughout the site. This is all done using Laravel routes.

Fig. 3 Register Page

A green screen with a black box

Description automatically generatedSimilar to the login page, the register page has everything rounded and the input boxes are enlarged to provide a sleek and simple design. There are error messages in place for checking matching passwords and whether the email is recognizable. A login button is provided in the bottom left corner so that users can easily direct themselves to the login page if needed. The links in the top right corner of the web page are still provided to users so they can easily navigate throughout the site.

Fig. 4 Reset Password Page

A screenshot of a computer

Description automatically generatedOnce a user selects “Forgot Password” they are redirected to the following page. This sleek design follows all the other pages on the site. All the user has to do is enter in their email and click “Send Password Reset Link” button. Laravel will check with the database to make sure the email is valid and then they will be set a password reset link to their email account.

A screenshot of a computer

Description automatically generated

Fig. 5 PC Builder Page

Once a user selects “Customize Your PC” they are brought to this page where they can pick each part individually. All items and prices are pulled from the MySQL database I have set up using Laravel controller. The prices all update automatically whenever a user changes a dropdown selection. The bottom also includes the subtotal, tax, and total. Subtotal being price before tax and total price with tax. Tax is calculated at seven percent of the subtotal as per Dorchester county law. All the prices are tracked and changed using JavaScript and local storage.

A screenshot of a computer

Description automatically generated

Fig. 6 Checkout Page

Using JavaScript and local storage I was able to create the checkout page to show the user what they selected and the price of each item as. All the prices are added up including tax to show the total price at the bottom. This acts as a receipt for the user. The home and logout options are still visible in the top right corner if the user wishes to place another order or logout once they complete their build.

Test Plan and Results

**Introduction**

This test plan shall ensure the functionality of the entire computer builder website from front to back. The three main parts of this website are the web scraping, database, and website. Each part will be tested thoroughly to keep the website running efficiently since each part depends upon the others. For the web scraping part, the goal is to test each individual function to verify that the data being pulled is up to date, relevant, and accurate. The goal of database testing is to check that the information obtained from web scraping is stored correctly and can be pulled efficiently by the website. While testing the website, the goal is to ensure that customers can easily navigate the website and purchase a computer with no hassle.

**Approach**

The following testing is designed to assess the reliability and speed of the website from front to back. These tests will include function calls, button clicks, and data insertion/deletion. The test method that will be used is white box testing because the tester knows all the internal components of the product. These tests will be executed manually due to multiple moving parts.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Test Case ID** | **1** | **Test Case Description** | Check database connection | | | |
| **Tester's Name** | | Nate | **Test Date** | 04/01/2024 | | |
| **S #** | | **Prerequisites** | | **S #** | **Test Data** | | |
| 1 | | MySQL Database | | 1 | Host = localhost | | |
| 2 | | Web-scraping Program | | 2 | User = root | | |
| 3 | |  | | 3 | Password = \*\*\*\*\*\* | | |
| 4 | |  | | 4 | Database = ComputerParts | | |
| **Step #** | | **Step Details** | **Expected Results** | | **Actual Results** | **Pass/ Fail / Not Executed /**  **Suspended** |
| 1 | | Launch web-scraping program | You are connected | | You are connected | Pass |
| 2 | | Wait for program to finish execution | You are disconnected | | You are disconnected | Pass |
| 3 | |  |  | |  |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Test Case ID** | **2** | **Test Case Description** | Check that all records are deleted from each table in the database | | | |
| **Tester's Name** | | Nate | **Test Date** | 04/01/2024 | | |
| **S #** | | **Prerequisites** | | **S #** | **Test Data** | |
| 1 | | MySQL database | | 1 | Table full of data | |
| 2 | | Web-scraping program | | 2 |  | |
| 3 | |  | | 3 |  | |
| **Step #** | | **Step Details** | **Expected Results** | | **Actual Results** | **Pass/ Fail / Not Executed /**  **Suspended** |
| 1 | | Run web scraper to delete all records in each table | Each table should be empty and no error | | No error | Pass |
| 2 | | Run MySQL query to show all records in each table | Each query should return empty set | | MySQL query returned “empty set” | Pass |
| 3 | |  |  | |  |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Test Case ID** | **3** | **Test Case Description** | Check that records are inserted properly into the according table | | | |
| **Tester's Name** | | Nate | **Test Date** | 04/01/2024 | | |
| **S #** | | **Prerequisites** | | **S #** | **Test Data** | |
| 1 | | MySQL Database | | 1 | Name (varchar 200) | |
| 2 | | Web-scraping Program | | 2 | Price (decimal(6,2)) | |
| 3 | |  | | 3 | table | |
| 4 | |  | | 4 | Insert into query | |
| **Step #** | | **Step Details** | **Expected Results** | | **Actual Results** | **Pass/ Fail / Not Executed /**  **Suspended** |
| 1 | | Run web scraper into query on table | Records inserted into table | | Records are inserted properly into appropriate table | Pass |
| 2 | | Run a select all query on table | New record should be in the table | | MySQL query returned all part names and price inserted | Pass |
| 3 | |  |  | |  |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Test Case ID** | **4** | **Test Case Description** | Check that main function runs on a 24hr timer | | | |
| **Tester's Name** | | Nate | **Test Date** | 04/01/2024 – 04/02/2024 | | |
| **S #** | | **Prerequisites** | | **S #** | **Test Data** | |
| 1 | | MySQL Database | | 1 | URLs | |
| 2 | | Web-scraping Program | | 2 |  | |
| 3 | | BeautifulSoup library | | 3 |  | |
| **Step #** | | **Step Details** | **Expected Results** | | **Actual Results** | **Pass/ Fail / Not Executed /**  **Suspended** |
| 1 | | Run main.py | You are connected | | You are connected | Pass |
| 2 | | Wait for disconnect message | You are disconnected | | You are disconnected | Pass |
| 3 | | Keep program running for 24hrs | You are connected, you are disconnected | | PC shutoff before 24hr timer | Suspended |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Test Case ID** | **5** | **Test Case Description** | When the submit button is clicked, the website reroutes to the checkout page | | | |
| **Tester's Name** | | Nate | **Test Date** | 04/01/2024 | | |
| **S #** | | **Prerequisites** | | **S #** | **Test Data** | |
| 1 | | MySQL Database | | 1 | Part names | |
| 2 | | Web-scraping Program | | 2 | Part prices | |
| 3 | | Website | | 3 |  | |
| 4 | |  | | 4 |  | |
| **Step #** | | **Step Details** | **Expected Results** | | **Actual Results** | **Pass/ Fail / Not Executed /**  **Suspended** |
| 1 | | Go to pcbuilder page | Pcbuilder page loads | | Pcbuilder loaded properly | Pass |
| 2 | | Select desired computer parts | Price is updated and names of selected parts is shown | | Price update failed and selected part was shown | Fail |
| 3 | | Click the purchase button | Checkout.blade.php is redirected automatically | | User is redirected to checkout page | Pass |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Test Case ID** | **6** | **Test Case Description** | Check that getProductInfo function pulls every item on each page only if it has a listed price | | | |
| **Tester's Name** | | Nate | **Test Date** | 04/01/2024 | | |
| **S #** | | **Prerequisites** | | **S #** | **Test Data** | |
| 1 | | MySQL Database | | 1 | URL | |
| 2 | | Web-scraping Program | | 2 | An empty array | |
| 3 | | BeautifulSoup library | | 3 |  | |
| **Step #** | | **Step Details** | **Expected Results** | | **Actual Results** | **Pass/ Fail / Not Executed /**  **Suspended** |
| 1 | | Call getProductInfo passing a URL as a parameter | The array is full of data where every even position in the array including 0 has is a string name and odds are prices stored as double | | Array is filled with part names and prices according to website they were scraped from | Pass |
| 2 | | Check that every iteration of the array has a price and part name | Array has no missing price or part name | | Every iteration of the array has one part name and one price | Pass |
| 3 | |  |  | |  |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Test Case ID** | **7** | **Test Case Description** | Check that runPages function will run recursively until there are no more pages | | | |
| **Tester's Name** | | Nate | **Test Date** | 04/01/2024 | | |
| **S #** | | **Prerequisites** | | **S #** | **Test Data** | |
| 1 | | MySQL Database | | 1 | URL | |
| 2 | | Web-scraping Program | | 2 |  | |
| 3 | | BeautifulSoup library | | 3 |  | |
| **Step #** | | **Step Details** | **Expected Results** | | **Actual Results** | **Pass/ Fail / Not Executed /**  **Suspended** |
| 1 | | Call runPages passing a URL as the parameter | Run pages calls itself (number of pages – 1) times | | Runpages ran 5 times with 6 pages | Pass |
| 2 | |  |  | |  |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Test Case ID** | **8** | **Test Case Description** | Check that getData function will return a string full of html code | | | |
| **Tester's Name** | | Nate | **Test Date** | 04/01/2024 | | |
| **S #** | | **Prerequisites** | | **S #** | **Test Data** | |
| 1 | | MySQL Database | | 1 | URL | |
| 2 | | Web-scraping Program | | 2 |  | |
| 3 | | BeautifulSoup library | | 3 |  | |
| 4 | |  | | 4 |  | |
| **Step #** | | **Step Details** | **Expected Results** | | **Actual Results** | **Pass/ Fail / Not Executed /**  **Suspended** |
| 1 | | Call getData function passing URL as a parameter | Returns a string containing all the html code from the requested URL | | Only requested HTML data was returned by getData function | Pass |
| 2 | |  |  | |  |  |
| 3 | |  |  | |  |  |
| 4 | |  |  | |  |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Test Case ID** | **9** | **Test Case Description** | Check that the price updates as user selects different parts | | | |
| **Tester's Name** | | Nate | **Test Date** | 04/01/2024 | | |
| **S #** | | **Prerequisites** | | **S #** | **Test Data** | |
| 1 | | MySQL Database | | 1 | Part names | |
| 2 | | Web-scraping Program | | 2 | Part prices | |
| 3 | | Website | | 3 |  | |
| 4 | |  | | 4 |  | |
| **Step #** | | **Step Details** | **Expected Results** | | **Actual Results** | **Pass/ Fail / Not Executed /**  **Suspended** |
| 1 | | Go to website | Website loads in browser | | Website is visible on local machine | Pass |
| 2 | | Change a part | Total price updates to reflect change in price | | Price was not updated | Fail |
| 3 | | Change a different part | Total price updates to reflect change in price | | Price was updated | Pass |
| 4 | |  |  | |  |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Test Case ID** | **10** | **Test Case Description** | Check that the website dropdowns populate with everything in the appropriate table | | | |
| **Tester's Name** | | Nate | **Test Date** | 04/01/2024 | | |
| **S #** | | **Prerequisites** | | **S #** | **Test Data** | |
| 1 | | MySQL Database | | 1 | Part names | |
| 2 | | Web-scraping Program | | 2 | Part prices | |
| 3 | | Website | | 3 |  | |
| 4 | |  | | 4 |  | |
| **Step #** | | **Step Details** | **Expected Results** | | **Actual Results** | **Pass/ Fail / Not Executed /**  **Suspended** |
| 1 | | Run web-scraping program | Database is updated with fresh data | | You are connected then you are not connected | Pass |
| 2 | | Go to website | Website loads in browser | | Website is visible on local machine | Pass |
| 3 | | Select a part dropdown menu | Every part listed in that corresponding table is shown to the user | | All parts in MySQL database populate the dropdown box | Pass |
| 4 | |  |  | |  |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Test Case ID** | **11** | **Test Case Description** | New users are able to register their account as long as email is valid | | | |
| **Tester's Name** | | Nate | **Test Date** | 04/01/2024 | | |
| **S #** | | **Prerequisites** | | **S #** | **Test Data** | |
| 1 | | MySQL Database | | 1 | Name | |
| 2 | | Website | | 2 | Email | |
| 3 | | Login Route | | 3 | Password | |
| 4 | |  | | 4 | Confirm Password | |
| **Step #** | | **Step Details** | **Expected Results** | | **Actual Results** | **Pass/ Fail / Not Executed /**  **Suspended** |
| 1 | | Go to register page | Website redirects to register page | | Redirected to register page | Pass |
| 2 | | Fill in input boxes | Errors pop up if applicable | | Bad email displayed error | Pass |
| 3 | | Click register | Account is created in database and website redirects to home page | | Account is created in database and user is redirected to home page | Pass |
| 4 | |  |  | |  |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Test Case ID** | **12** | **Test Case Description** | Returning users are able to login if account is already created | | | |
| **Tester's Name** | | Nate | **Test Date** | 04/01/2024 | | |
| **S #** | | **Prerequisites** | | **S #** | **Test Data** | |
| 1 | | MySQL Database | | 1 | Email | |
| 2 | | Website | | 2 | Password | |
| 3 | |  | | 3 |  | |
| 4 | |  | | 4 |  | |
| **Step #** | | **Step Details** | **Expected Results** | | **Actual Results** | **Pass/ Fail / Not Executed /**  **Suspended** |
| 1 | | Go to login page | Website redirects to login page | | Website redirects to login page | Pass |
| 2 | | Fill in input boxes | Errors show if account isn’t in database. Otherwise no errors. | | No errors | Pass |
| 3 | | Click Login | Account is verified and user is redirected to home page. | | Account is found and website redirects to home page | Pass |
| 4 | |  |  | |  |  |

Challenges Overcome

**Are You a Robot?**

The first and biggest challenge I faced was being spotted as a bot when web scraping Amazon or Newegg. If I sent too many requests too often or from the same computer using the same header, their servers would deny my request for the HTML. To get around this I began using some rotating headers and making sure I didn’t run the web scraper more than necessary. It also helped to swap between web scraping from Amazon and Newegg instead of doing either multiple time in a row.

**Laravel**

I first began the website using the PHP language itself within my HTML code. However, this was a headache and very time consuming. That is why I decided to use Laravel. Little did I know though that Laravel has a steep learning curve. Learning how Laravel works, and its syntax was also time consuming. The most important and difficult areas about Laravel are models, controllers, and routes. Though once you learn them, they become powerful tools to enhance any website.

**JavaScript**

This project required me to use more JavaScript than I ever have. I had to learn how to use action listeners for when dropdowns were changed. I also utilized local storage to pass the prices and names of the parts to other pages such as the checkout page. Another difficult challenge was the math. I had to try probably 30 different combinations or ways to round the prices to 2 decimal places before I found out how to properly do it.

**Future Enhancements**

In the future I would like to implement an AI component that can recommend users a PC build based on a few questions. The questions would at least ask about the user’s budget, use for the PC, and gaming. Then it would populate the dropdowns with parts based on the questions and within the user’s budget. I would also like to increase the number of websites I scrape from so that I can ensure users get the best deals possible. Lastly, I would like to use a payment processing API to accept payments through the website without having to have direct contact with customers for payment.

**Defense Presentation Slides**

A black background with white text

Description automatically generated

A black and white background with white text

Description automatically generated

A black and green background with white text

Description automatically generated

A screenshot of a computer program

Description automatically generated

A green hexagons with white text

Description automatically generated

A screenshot of a computer game

Description automatically generated

A green screen with text

Description automatically generated

A close-up of a question

Description automatically generated