



# **J-Tech Ecommerce Website**

<b>Analysis</b>	<b>4</b>
Problem identification	4
Stakeholders	5
Why is it suited to a computational solution ?	6
Owner Interview	7
Interview	8
Analysis	8
Similar existing e commerce websites	9
PC Specialist	9
Overclockers UK	9
AWD-IT	10
Features of proposed solution	11
Concept of my solution using my research	11
Limitations	11
Further contact with owners	12
Email	12
Response from owners:	12
Requirements	12
<b>Design</b>	<b>12</b>
Functionality	13
Hardware and software	14
Success criteria	14
<b>Design</b>	<b>15</b>
Introduction	15
What are the problems	15
Homepage design	17
Product page design	19
Cart page	20
Checkout page	21
Login Page	22
Automated order confirmation email design	22
Database	23
Flowcharts	24
Client-server architecture	24
Login	24
Adding product to cart	26
Test Data	27
Validation	27
Iterative Development Test Data	27

<b>Implementation</b>	<b>28</b>
Test data	28
First iteration of Solution	29
Main Javascript File	29
Images of website prototype	45
Home page	45
Product page	45
Single product page	46
Cart page when not logged in	46
Cart page when logged in	47
Checkout page	47
Login page	48
Register page	48
<b>Evaluation</b>	<b>49</b>
Objectives	49
Primary objectives	49
Maintenance issues	50
Limitations	50
End user Feedback	50
Owners response	50
Employees	51
Customer opinion (Focus Group hosted by J-Tech Marketing Team)	51
Usability	51
Future Improvements	52
Conclusion	52
Home page	69
Product page	74
Cart page	78
Single product page	81
Search page	84
Login page	87
Register page	89
Checkout page	91
Order Confirmation	94
Script file	96
CSS files	98
Cart css	99
Checkout css	105
Login css	107

Product css	111
Register css	114
Main CSS file	117

## Analysis

### Problem identification

I am working with an upcoming company called J-Tech that sells computer components and prebuilt computer systems with the latest components. They have requested for a website to be made for the company as they need to have a large market available to be able to sell to. By having a website it will expand their company globally allowing for sales to be made across the globe. This website will have a products page, home page and a cart page. They have asked that the site has a modern and professional look with a top selling products section that uses data from the most purchased products.

The website will have to be coded in the usual HTML, CSS and JavaScript to have interactive elements within the site. A database will be needed for the site to store the data of customers such as their login details, shipping information and payment method. This will have to be kept secure through a variety of different security methods so we do not breach the data protection act. The database will use queries to get information such as the top selling products as well as using the database to store products so when the company wants to add a new product they do not have to make a hard copy of the html product page. This will make the site more dynamic for the users and for J-Tech allowing for better methods to be used to store the information about customers and products rather than having to make a new html page each time.

Depending on the amount of data that needs to be stored can make the cost of the project a bit more as they may have to purchase a server to store all of the data onto as if they get hundreds of new customers a day they will have lots of data to store on the server. Another factor is that the amount of products they are entering into the database will increase the amount of storage it takes. This would be down to the amount of storage it takes to store product images. The database that will be used to store the products of the websites will have an admin control section so anyone with the admin details can add the product details to the website with ease. This will ask for the admin to upload the product image, description, name and price this will already be formatted through the CSS code so when the customers of the site click on the product they wish to buy the page will be made automatically getting rid of the need for having hard coded product pages each time.

The addition of a website to J-Tech's business model brings valuable benefits, allowing the company into the global marketplace while enhancing customer convenience and satisfaction. With the ability to reach customers worldwide, capitalise on online sales

opportunities, and gather insightful data for strategic decision-making, the website becomes a cornerstone of J-Tech's growth strategy. The cost efficiencies and scalability offered by an online platform ensure long-term viability and competitiveness in e-commerce.

## **Stakeholders**

I am working with an upcoming business called J-Tech whose sales team reached out to me to develop a fully working website for the company. They have not stated any other external stakeholders within the project apart from the sales team and the company. I will have to conduct some form of research for the stakeholders to get a greater understanding of what they want to see within the website and what the layout should look like. I will ask them to draw me some wireframes of what each page of the site should look like so that I can meet their designs for the site as quickly as possible.

This benefits the company overall as by having a website for your products and selling them online expands your market at a large scale allowing for the more profit to be made. My website will allow the upcoming customers of J-Tech to purchase products from anywhere across the globe. This means the company can potentially grow more and become more well known as people can purchase products from anywhere. This system will improve the overall customer experience and also adapt to modern changes in life as of now not many companies open up physical stores to make it easier for customers to purchase products from the comfort of their own home.

When the product is ready to go live the staff of J-Tech will have to learn how to use the new website so they can know how to update it with new products and how to see the orders being placed so they know what to start shipping out to who and what address. This will all be logged onto a database so all information will be stored there so if at any point a customer has a problem with their order the staff can go back to the database and find their information to help solve the problem. The staff of J-Tech will be the ones who use the system on a daily basis as they will need to update it with the products they are selling and the quantity they have in stock of the product this is so when a customer places an order they don't pay for something that is out of stock keeping customer satisfactory level to a high making a greater brand image for J-Tech.

This benefits customers for J-Tech as this allows for products to be more available and not require them to go to a physical store to get the products they want. This increases the global scale the j-tech reaches and this is good for the customers as they have more products available to them this will drive them towards purchasing from J-Tech and customer satisfaction will increase.

### **Why is it suited to a computational solution ?**

This project is suited to the company as rather than them having to spend money having physical stores in different locations that not many people will go into to buy from the website will expand their potential market to a larger scale allowing for more sales to be made across the globe generating more money for them. This will help the company become more recognised among the other companies selling computer components as not many physical stores will not be able to survive as most customers purchase products online.

I will use a range of HTML, CSS, Javascript and node js to code the website i have done some basic work with html and css and i will be learning more into node and Javascript as i dont have the best understanding of them and i want to further develop my knowledge. I will be using javascript to make interactive and dynamic aspects of my website, and node js will be used for the database within the site to contain all the products and make product pages with them. I am going to use these coding languages as I have researched them and most websites will use coding languages such as these and this will help further develop my coding knowledge.

The advantages of using these languages is it will make the website more dynamic and interactive. This means the website will seem more professional and will be accessible to the customers of the website allowing them to do features such as adding products to the cart. Another advantage of me using these languages is that I have a better understanding of them compared to other languages that might be used. One disadvantage of this project is that I've never built something this complex and at this scale so I will need more time to learn and research into the most effective method of doing so.

If I wasn't on such a limited time scale I would be spending more time learning other methods of doing features of the website to make it perform better and go for more elements used within the site. This could be features such as an interactive map that you see on other sites or a login system. As i dont have the time and resources I won't be able to do these features and will have to stick to a simple dynamic site.

I think this project is well suited to be a computational solution as it will increase the quantity of sales for the company allowing for their brand to become more recognised as well as it saves them money having to make physical stores in locations. This also helps to expand the potential market they can conquer as being online and being able to deliver to anywhere across the globe allows for sales to be made at a larger scale. This system will be simple for customers to use as if they have a simple understanding of how e-commerce websites work they will have no problem navigating the site to purchase the products they wish to purchase. The website will include a responsive design so it should be capable of fitting on any device so no matter what device a customer is using they will be able to purchase products. All sales made will be stored in a database and this will contain their

shipping information and card details. If J-tech ever needs to update their products page they will be able to do so by a page that will only be accessible to them which will display different information required for the products to be added to the page this will be things such as product name, product price and product description.

## **Business analysis**

Having a website helps businesses analyse data about how people use their site. By tracking things like what pages they visit, what they purchase, and what content they like, businesses can learn a lot about their customers and improve their website and products they sell to attract more people and sell more stuff. Websites also let businesses ask customers for feedback and see what their competitors are doing, helping them stay competitive and make smart decisions.

## **Owner Interview**

I will be asking the owner of the company a series of questions on what they would like the system to include. I will be using the answers from this interview to get the requirements he wants for the site, what needs to be included in the site and get a better overall understanding of how he wants the site to function.

The questions I am asking him are:

- 1) What pages do you wish to have within the site?
- 2) Why do you want to have a website over a physical store?
- 3) Are there any mistakes that could potentially occur from having this website if so what are they and how can we get around fixing them.
- 4) What does the site need to be able to do effectively?
- 5) Is there anything specific you would like to see within the site?
- 6) Are there any specific payment options you take?
- 7) Are there any questions you would like to ask me?

The first question is intended to help me find out if there are a specific amount of pages he would like to have and what pages he would like within the site. The second question is to help me get a better understanding of why he would want to have a site over a physical store in case the website does not launch as intended.

Question three is to help me find out what potential problems they may face from the system such as human error occurring and causing orders to be missed or not done properly. Question four is asking what is the overall aim for the website so i can build the overall product to what he wants it to do effectively.

Question five is asking if they would like to have any specific feature in the site such as a map or a contact page feature. This lets me get a better understanding on the complexity level of the project. Question six is me checking to find out if I have to display within the

site that they only take a specific payment option such as paypal or only visa and mastercard.

Question seven is me seeing if he has any questions for me about the project so I can help him with any queries he has.

## **Interview**

1) What pages do you wish to have within the site?

"The website should have a about us page, home page, product page, and contact us page"

2) Why do you want to have a website over a physical store?

"By having a website it will increase number of sales we can potentially generate as we will be operating at a larger scale"

3) Are there any mistakes that could potentially occur from having this website if so what are they and how can we get around fixing them.

"If employees are not focusing when packing orders items can be missed so if you can find a method that sends a list of items to the employee and the shipping address customers should be able to get what they purchase."

4) What does the site need to be able to do effectively?

"Overall the site needs to be able to sell products to the customers with ease."

5) Is there anything specific you would like to see within the site?

"The main feature I would like to see is a contact us feature if that is possible for you to do this can be a page with our email or phone number on allowing customers to reach us if they have any issues with purchases."

6) Are there any specific payment options you take?

"We take all payment options such as visa, mastercard, paypal"

7) Is there anything else you would like to add?

"As Long as the system is clean and professional design and employees are able to add products to the site it will be good."

## **Analysis**

This has given me a better understanding of what I need to be doing for the website to meet the requirements they have set in this interview. This allows me to now do a graphical mockup of the website so I can start developing a prototype of what the end



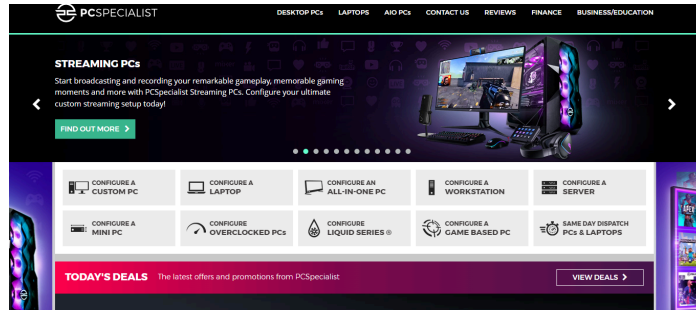
Jamie O'Neill

product will look like. The company will have to spend some money getting product photos done and editing them for their website so that they don't breach the copyrights, design and patents act, this means they will have to either learn how to edit photos or hire someone to do this and pay for editing software. I need to make the database first so I can make the product page dynamic and work off of that so I don't hard code html pages with products into the website.

## Similar existing e commerce websites

### PC Specialist

PC specialist is a pc ecommerce website that sells a range of different products all to do with computers. The site has a very unique design that most other websites don't have which makes it memorable and sticks out compared to other ecommerce websites. It has a rotating banner that shows special promotions, sales and new products, this makes the site seem overall and is a nice feature to have. It is easy to purchase the products cause all you gotta do is click on your account and enter your card details. This website also has a custom PC builder which allows for the users to configure the parts to build the PC they want and from that they can have a pc specialist build ii in case they don't know how to or do not have the time to.

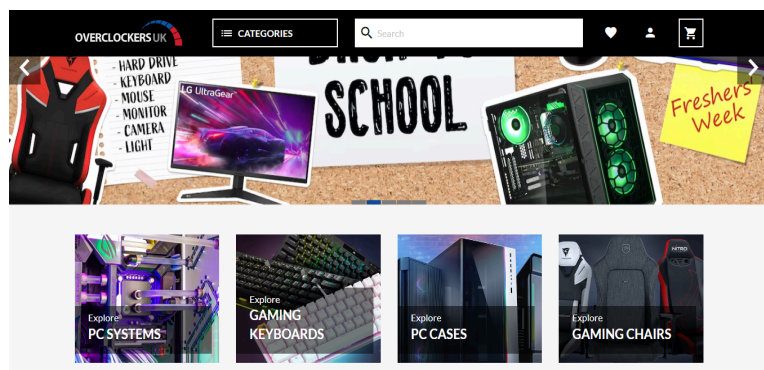


### Aspects I can take to for my project:

The overall design and layout sticks out so I might take some elements of it as inspiration for my website. This will include features such as the rotating banner and buttons within the site. When the customer makes a purchase on the site it emails a receipt to the end customer and also shows on the website once the product has been purchased displaying the products they purchased and the price of each item with the total and the shipping address. The receipt is sent to the customer's email in a pdf file containing information about the purchase.

### Overclockers UK

Overclockers website has a similar design to PC specialists with the banner. However this design is more unique as the buttons to other pages are



images which makes the website overall more appealing. However this site can be confusing to navigate if the user doesn't understand how they have laid out the site. When you go to purchase your items on the website it is a very easy process and not long to do compared to other sites where you have to go through multiple pages for each bit of information they request from you. When viewing the product the product page is a clean and simplistic design and only contains important information such as price, name and description of the product.

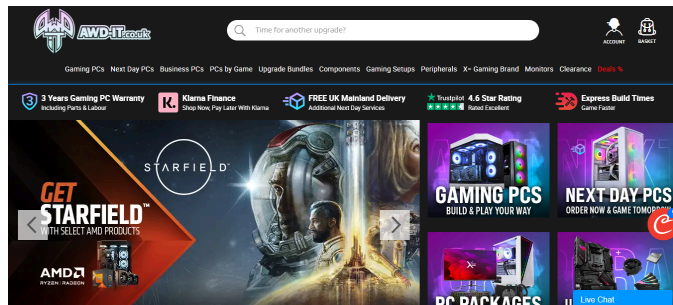
### **Aspects I can take to for my project:**

I would like to see how they have done the checkout page as it is more efficient and friendly to the user rather than them having to navigate from one page to another to purchase products. I like the way they have presented the product page as it is simple and not too overloaded with useless information which is vital to have for the website. This makes it easier for customers and more likely to promote sales if it is not over cluttered with information as it's easier to add to cart and they don't waste time reading useless information that is not needed.

### **AWD-IT**

AWD-IT is another computer e-commerce website that uses a simple easy with a brief description of what the product type is for example the gaming PCs section says "Explore high-performance gaming PCs and custom gaming

PCs for an immersive gaming experience. Upgrade to the ultimate PC gaming setup now with the UK's leading custom gaming PC builder!". This gives the customer more explanations of what they might be potentially buying. This also shows and informs the customer about a feature they have on the website called the PC Builder. The layout of the website is a very cool and unique design; it incorporates a rotating and interactive banner that shows deals they have on their site. This banner is scaled down and next to it it shows a button which directs the user to the page for that type of product. I overall really like the design, layout and colour scheme of the site which makes it stand out more compared to other sites.



### **Aspects I can take to for my project:**

Some aspects I like about this website are the layout, colour scheme, navigation bar and logo placement. This website is very easy to purchase products from, however finding and navigating the site is difficult as there are so many pages within the site causing confusion when searching for certain products. I will most likely consider taking the colour scheme from the site as that overall makes the site more visibly appealing and more likely to keep a user using the website as if the website had a bad design visually or looked outdated it would not be successful at selling products as customers would not trust it.

### **Features of proposed solution**

#### ***Concept of my solution using my research***

My project involves creating a web-based site using HTML, CSS, JavaScript, Node.js, and MySQL. These technologies were chosen to meet the requirements of building a dynamic and user-friendly product page. To achieve this, I plan to use EJS pages for dynamic content, ensuring that new products can be easily added and updated on the site. This approach will require me to further develop my understanding of these languages, particularly in the context of web development. The unique design of the site is crucial to attract users and differentiate it from other similar websites, ultimately convincing users to choose our platform for purchasing products. The chosen software stack, including HTML, CSS, JavaScript, Node.js, and MySQL, is well-justified for the project. These technologies are essential for creating a dynamic and user-friendly website. EJS pages will be used to automate product updates, meeting the project's requirement for easy product management. XAMPP will be used as a local development environment to run the web server known as Apache, database server MySQL, and interpreter for server-side scripting language PHP. XAMPP is essential for testing and debugging the website before deploying it to a live server as it allows me to see the preview of the site and test the server functionality.

#### ***Limitations***

One limitation of this website is I have not made some of the features that I would like to incorporate before. As of this I will need to learn how to make these and this will cost me more time that I could be spending on this project. I want to make it so the customer gets emailed a receipt to their email. However, I don't have any idea how this works so I will need to research this so I can make the system seem more professional and incorporate this feature. Another limitation with making this site is that there needs to be a payment system so users can purchase products that will need to be in the checkout section of the site. I'm not so sure on how you make one of them and as a result this will need to be looked into and I need to implement it into the system. I have never published a website

Jamie O'Neill

before and because of this I will need to find out how this process is done and whether or not the domain name is already taken as if so I might not be able to use it for the company.

### **Further contact with owners**

I sent an email to the people incharge of making the website and made a simple list of what requirements will need to be met that I have found from the research I have conducted.

#### ***Email***

“Hi,

I have been researching more into the project you have set me for J-Tech. I have conducted a large amount of research and from this I have come up with a list of requirements that are needed for this project that I am in the steps of developing for you and I wanted to double check with you if this is what you are looking for. The requirements i have come up with are:

- Unique and and easy design for the website
- Easy to navigate
- Checkout system
- A home page, product page, contact us page and checkout page
- Top selling products on the home page

Many thanks

Jamie “

### **Server requirements**

The server will require certain things for the server file to work

- Xampp (Apache and MYSQL)
- Nodejs
- Ejs
- Bcryptjs
- Mysql
- Express
- Html
- Css
- PHPMyAdmin

***Response from owners:***

“Hi Jamie,

I reckon that the set of requirements is well suited and well developed enough for this project. The only thing we want to add on is a method to add products to the site as this will be needed to update the site with new products. “

**Requirements**

**Design**

Requirement	Explanation
Unique and easy design	This is so the website stands out against other websites that sell the same products and the design needs to be easy to navigate so it is not confusing for the user.
Easy to navigate	This is so the users can purchase products with ease and not get lost within the website as some websites are hard to navigate to find the particular thing you want.
A home page, product page, contact us page and checkout page	These pages are the fundamentals of the website as without these it would seem like a unfinished project and if there was not a product page it would become more confusing for the user if all the information was on one page and not layered or organised into sections this help split information and products into pages so one page is not so over cluttered.

## Functionality

Requirements	Subsystem	Explanation
Menu	Takes you to another page or section on a page	This is so the user can get to another page or section easily with this button it stops the navigation bar being overloaded.
Checkout	Total for sale	This gives the price off all the products the user have added to the cart and totals them to get the payment charge correct
	Enter payment details	This is to secure the transaction of the products and get the user payment details to charge them for the products.
	Shipping address	This is to get the information from the user so J-tech can deliver the product to the customer making them ecommerce company.
Stock	Update number of stock	This is so no customers purchase any products that aren't in stock.

## Hardware and software

Requirement	Explanation
A computer or laptop with a keyboard mouse	The user needs to be able to enter products into the database to make the website be up to date and correct with the products available. This needs to have enough storage to hold the data as the database later down the line it may get bigger in size.
Window or mac or linux operating system that is capable of running javascript.	This operating system needs to be able to run javascript as the database will be made using sql and node js for better productivity.
A server to host the website on	A web server can either be made by yourself depending on the skills you have or can be purchased online for hosting the website on the world wide web. This can be done on websites such as hostinger which offer different hosting packages for different type of websites.
Need XAMPP	J-tech will require to have XAMPP installed on the laptop desktop that the products will be added on this is so products can be added to the website.

### Success criteria

Requirement	Explanation
Customer need to be able to purchase products	This is required as J-tech needs to take the payments from the customers otherwise payments may be missed or not collected. This can be done through paypal who have a feature available to be added to a website.
Customers need to be able to search products	Customer need to be able to search for the products they are looking for as the product page might be full of 1000 or more products which means they will be searching for ages.
Employees need to be able to add products to the website through a database .	This is a must have as if customers have to hard code the product to the page it will become an issue as the file size will get bigger and the employees doing will not know how to. This will be done through some sort of database being made and it will ask for information such as product name, price, image and description
Need to be able to add product to cart	This needs to work so customers do not end up having to purchase each product one at a time. This will be a page that has a list of products and their price with an image and a total at the bottom. This is vital to have as this is the main function of the site.
Need to be able to total the price in the cart section	The total price needs to be shown once all items are in the cart and are ready to be purchased.

# Design

## Introduction

In my website I need to focus on main aspects to make sure the end customer is happy with the end result. The product page loads products through a database that the company can access with XAMPP. This will dynamically load products into the page rather than them having to be hardcoded into the website. The users need to also be able to search for the products they wish to purchase rather than navigating the whole site to find a product. Another aim is to have the site have a fully functional and interactive design that works and looks professional to the end user. The staff will need to be trained on how to add products to the database as it is not that straightforward of a process. A login page will need to be made so when an order is placed an email is sent to the customer purchasing the products.

## What are the problems

To make this project easier for me when coding it I will have to break down each problem into smaller problems. This will allow me to be doing one step at a time rather than going head on with the project as I may forget aspects that are needed to be done. As soon as I have completed a step I will have to test the website to see if it is working with the new thing I have added. For example this could be things like making sure the database is loading the products correctly without any issues so the website can function properly. This will be good for the website as if i test it at the end once im done with coding the whole thing and i have a large list of problem with the code i will be spending a lot of time potentially deleting aspects that are not needed for project and wasting time and not using the time i have the best i could this will also mean that the main aspects will work and there will only be small minor bugs that i can fix quickly. This is an example of agile programming and this is probably the most efficient method for me to program with as the others potential ways I could approach this would not be as efficient. These are the problems i will be breaking my project into:

- Database
- Web design layout
- Sending automated email to customer with purchased products
- Home page

These problems will have smaller problems within them which will be a challenge when it comes to developing these areas of the project. By having the problems made into little problems it will save time as I will be working on one small problem at a time. The smaller objective will help me to focus on the main aspects of the website same with the most important areas.



Main Area	Problems	Why is this important
Database	Needs to upload products properly to the website with the css styling working right.	This needs to work 24/7 as if it is not working and the styling is not working with the database text appearing and the website will look really unprofessional.
Web design layout	The layout will need to look professional and needs to cooperate with the database as issues may arise.	If the web design is not professional or does not work how it is meant to, the client will not be happy especially if the database text is just appearing and not being styled within the website.
Sending automated messages to customers when they have purchased products.	The messages may be sent to spam or not be received by the end user	If the email is sent to spam the customer may not be in on the day the delivery is stated and the order may cause delivery issues and unhappy customers.
	The email may not consist of the right products or information.	This is very important in case the customer is missing something from their order and they have a receipt to prove it so no orders or products go missing.
	The email address needs to be validated in case they enter a random thing and not an actual email address.	If the email address is not validated the user may never get the email and this will cause issues as the email is not being formatted and stored correctly so this will cause issues with the delivery of the receipt of the order.
Home page	The homepage layout and design needs to be user friendly and cooperate with the database. This also needs to have a decent amount of information and products on it all style it appropriately.	If the design is not responding with the database and the website homepage is overcrowded with information that is not needed and not visually appealing to the user and may deter users from using the site.

I think that these are the vital points of the website that I have mentioned and need to be brought up to the company with what they want done for the website. I think these are important as this was mentioned in the interview on how they need to be able to add products to a database that links to the site. This increases the functionality of the site and makes it so they don't have to ask someone else to make the product as that will become high in cost for them to do and they don't have the skills to do so. All of these points that have been highlighted are vital for the website to succeed especially as the website would be a waste of costs for the company if the company didn't have these features. This is the main objective I need to focus on as this guarantees the success of the website once it is published. These are also a guide for me so that I don't add aspects that are not needed for the website. This is why I have chosen to use iteration in this project as this will help with the development of the prototype for J-Tech.

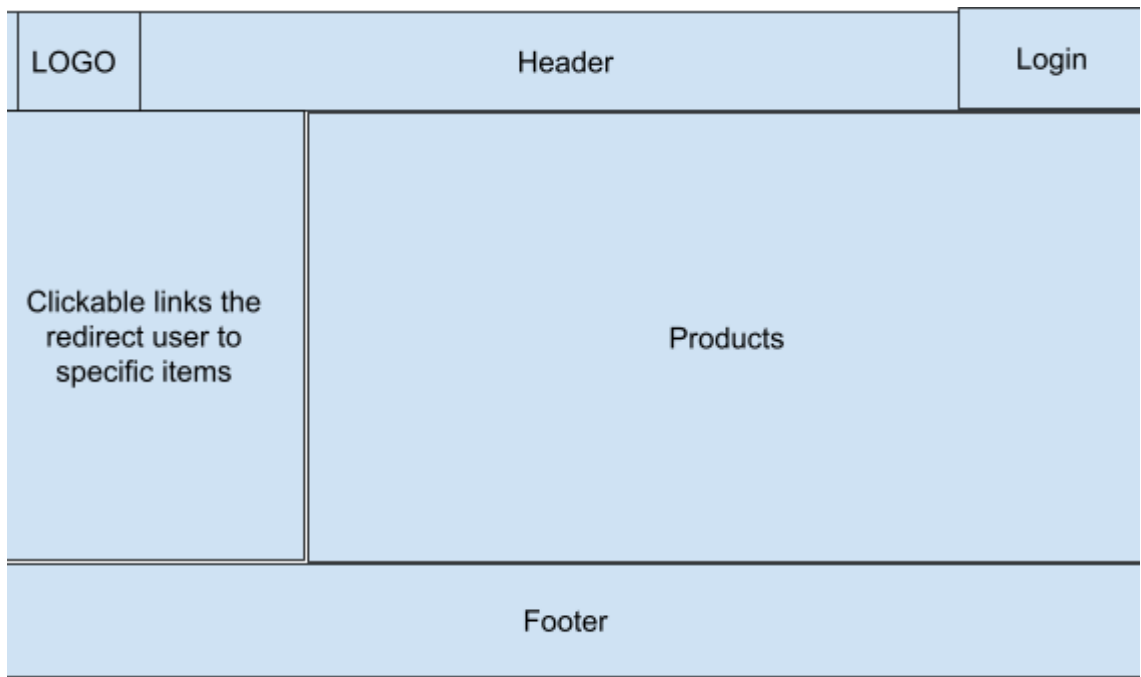
## Homepage design

This will be what I aim to make my system look like as this is the style and structure of other computer selling websites that are similar to what the main project is. Using the requirements and objectives I have now got I will meet the clients aims.

LOGO	Header	Login
Navigation Bar		
Advertise New promotional offers		
Top Selling products		
Sitemap		
Footer		

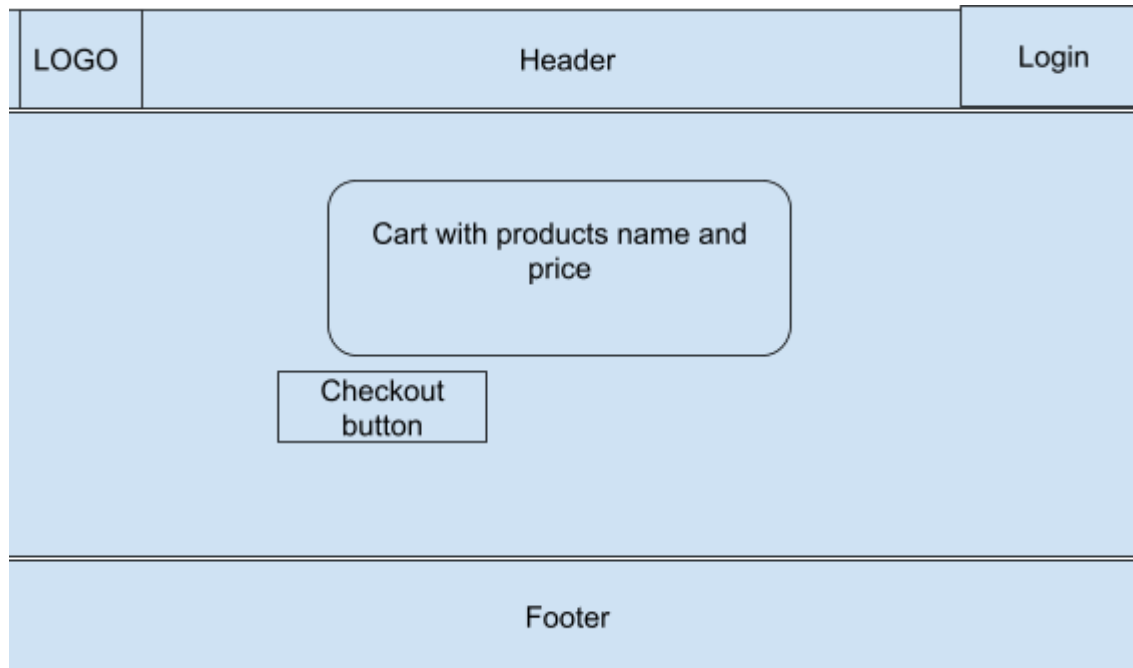
This is a wireframe of what the homepage will look like; it will have the navigation bar at the top with the logo at the top left which is more suitable for a website as most sites all follow a similar layout and it is what customers are more used to. There will be a banner underneath the navigation bar that will show advertisements and will rotate on a time scale. This is to promote certain products to users and by having it placed at the top of the page it stands out to the users. There will be a top selling products section that shows the top selling products that will be done by a SQL statement that searches for the products with the least amount in stock.

**Product page design**



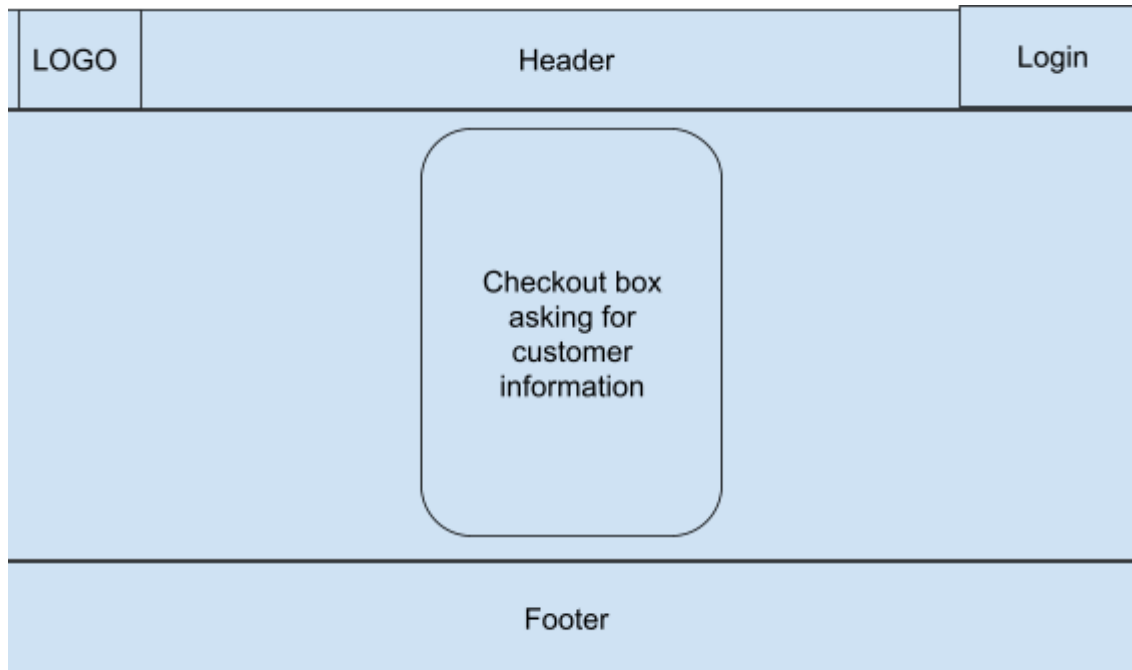
The product page will have products that will be loaded by a database to make load times faster and more suitable for this specific design with it being dynamic with clickable links to the different categories of products. This will follow the same layout as the homepage and style and when the product is clicked on it will create the page for the products for optimal performance.

## Cart page



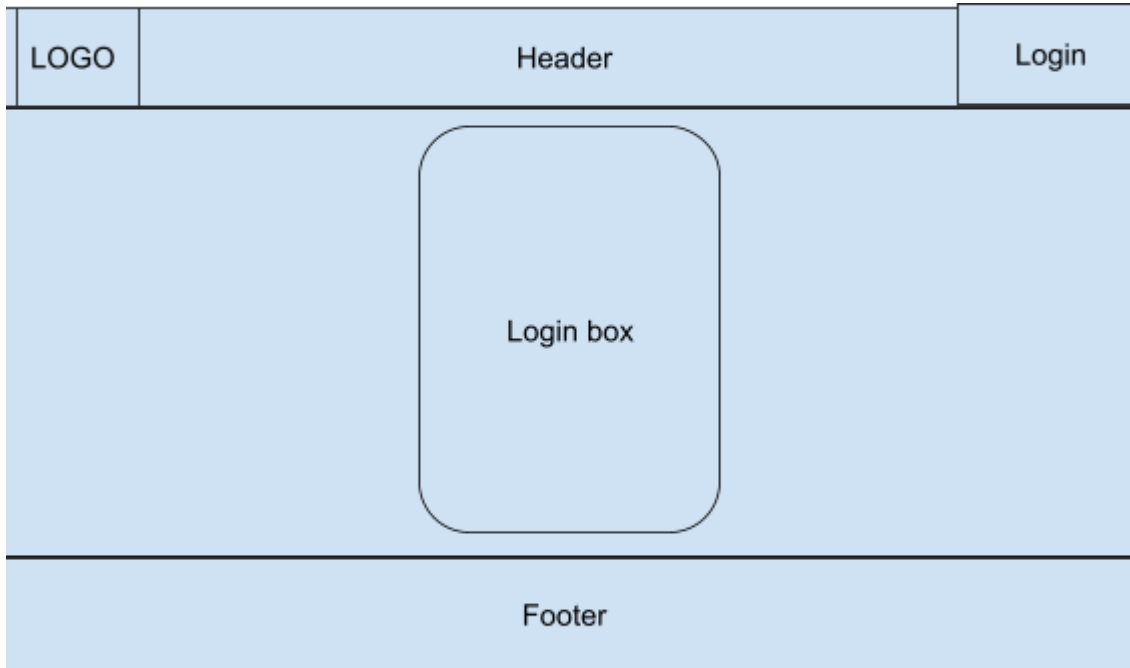
The cart page is designed to be user friendly allowing for it being easy for the user to see what they are buying as it is laid out in a list with the product name and price with the total at the bottom which is a method that all the ecommerce sites use to show the products someone is ordering by having it in a clear layout.

## Checkout page



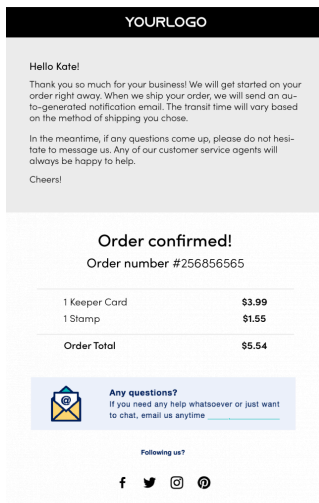
The checkout page is designed to get all the users information in a clear and simplistic layout that other Ecommerce sites use. This design is friendly to the users and is clear and simple when they are asked for their information.

## Login Page



The login box is a little box that is in the same style as the rest of the pages within the site and this box links to the database that stores their information. This box request for the username or email address and password.

## Automated order confirmation email design



The confirmation email will have a design similar to most pre-existing companies as the layout and design is the best for confirming the order. It will have a short paragraph saying thank you for shopping at J-Tech and underneath that will be the order number and the list of products with total price paid.

This will be sent automatically once the payment for the order has gone through and the user will also receive another email after stating that it has been shipped and is out for delivery. This is vital for the website as some of the products they will be selling are not cheap and are quite costly.

## Database

The database will be accessible to the employee who is adding products to the website through PHPMyAdmin. For this to be accessible the user needs to have Xampp running on their laptop or desktop to access the php page once that is running the employee will be able to add products to the page by accessing the phpmyadmin website. The employee will require some training on how to add the products. This is what the screen will look like for the employee adding the products.

**phpMyAdmin** [Database: test] [Structure: user] [General]

[Browse](#)
[Structure](#)
[SQL](#)
[Search](#)
[Insert](#)
[Export](#)
[Import](#)
[Operations](#)
[Triggers](#)

#	Name	Type	Collation	Attributes	Null	Default	Extra	Actions
1	id	int(11)		None	NO	AUTO_INCREMENT		<a href="#">Change</a> <a href="#">Drop</a> <a href="#">Browse</a> <a href="#">Distinct values</a> <a href="#">Primary</a> <a href="#">Unique</a> <a href="#">Index</a> <a href="#">Spatial</a> <a href="#">Fulltext</a>
2	id_rsa	blob(200)		None	NO			<a href="#">Change</a> <a href="#">Drop</a> <a href="#">Browse</a> <a href="#">Distinct values</a> <a href="#">Primary</a> <a href="#">Unique</a> <a href="#">Index</a> <a href="#">Spatial</a> <a href="#">Fulltext</a>
3	password	text	utf8_general_ci	None	NO			<a href="#">Change</a> <a href="#">Drop</a> <a href="#">Browse</a> <a href="#">Distinct values</a> <a href="#">Primary</a> <a href="#">Unique</a> <a href="#">Index</a> <a href="#">Spatial</a> <a href="#">Fulltext</a>
4	author	text	utf8_general_ci	None	NO			<a href="#">Change</a> <a href="#">Drop</a> <a href="#">Browse</a> <a href="#">Distinct values</a> <a href="#">Primary</a> <a href="#">Unique</a> <a href="#">Index</a> <a href="#">Spatial</a> <a href="#">Fulltext</a>
5	booktype	text	utf8_general_ci	None	NO			<a href="#">Change</a> <a href="#">Drop</a> <a href="#">Browse</a> <a href="#">Distinct values</a> <a href="#">Primary</a> <a href="#">Unique</a> <a href="#">Index</a> <a href="#">Spatial</a> <a href="#">Fulltext</a>
6	price	float(11)		None	NO			<a href="#">Change</a> <a href="#">Drop</a> <a href="#">Browse</a> <a href="#">Distinct values</a> <a href="#">Primary</a> <a href="#">Unique</a> <a href="#">Index</a> <a href="#">Spatial</a> <a href="#">Fulltext</a>
7	stock	int(11)		None	NO			<a href="#">Change</a> <a href="#">Drop</a> <a href="#">Browse</a> <a href="#">Distinct values</a> <a href="#">Primary</a> <a href="#">Unique</a> <a href="#">Index</a> <a href="#">Spatial</a> <a href="#">Fulltext</a>
8	prefversion	int(11)		None	NO			<a href="#">Change</a> <a href="#">Drop</a> <a href="#">Browse</a> <a href="#">Distinct values</a> <a href="#">Primary</a> <a href="#">Unique</a> <a href="#">Index</a> <a href="#">Spatial</a> <a href="#">Fulltext</a>
9	address	text	utf8_general_ci	None	NO			<a href="#">Change</a> <a href="#">Drop</a> <a href="#">Browse</a> <a href="#">Distinct values</a> <a href="#">Primary</a> <a href="#">Unique</a> <a href="#">Index</a> <a href="#">Spatial</a> <a href="#">Fulltext</a>
10	allowed_publishers	text	utf8_general_ci	None	NO			<a href="#">Change</a> <a href="#">Drop</a> <a href="#">Browse</a> <a href="#">Distinct values</a> <a href="#">Primary</a> <a href="#">Unique</a> <a href="#">Index</a> <a href="#">Spatial</a> <a href="#">Fulltext</a>
11	book_category	text(255)	utf8_general_ci	None	NO			<a href="#">Change</a> <a href="#">Drop</a> <a href="#">Browse</a> <a href="#">Distinct values</a> <a href="#">Primary</a> <a href="#">Unique</a> <a href="#">Index</a> <a href="#">Spatial</a> <a href="#">Fulltext</a>
12	categories	text	utf8_general_ci	None	NO			<a href="#">Change</a> <a href="#">Drop</a> <a href="#">Browse</a> <a href="#">Distinct values</a> <a href="#">Primary</a> <a href="#">Unique</a> <a href="#">Index</a> <a href="#">Spatial</a> <a href="#">Fulltext</a>
13	cities	text	utf8_general_ci	None	NO			<a href="#">Change</a> <a href="#">Drop</a> <a href="#">Browse</a> <a href="#">Distinct values</a> <a href="#">Primary</a> <a href="#">Unique</a> <a href="#">Index</a> <a href="#">Spatial</a> <a href="#">Fulltext</a>

[Check All Tables All With selected](#)
[Browse](#)
[Change](#)
[Drop](#)
[Primary](#)
[Unique](#)
[Index](#)
[Spatial](#)
[Fulltext](#)

[Print view](#)
[Propose table structure](#)

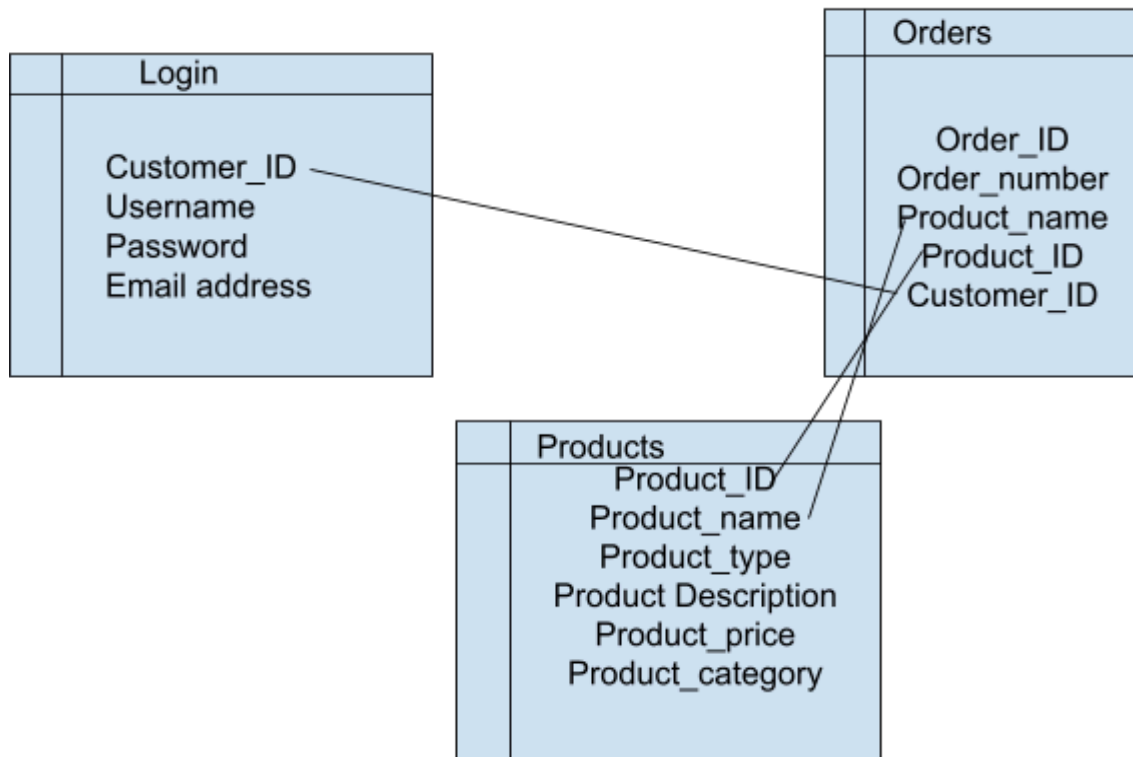
Add:  columns to:  At End of Table  At Beginning of Table  After  Before

[Indexes](#)

**Information**

Space usage	Row Statistics
Data: 217 MB	Format
Index: 210.3 MB	Collation
Total: 427 MB	Rows
	106,348
Row length	
	799 B
New statistics	
	106,348
	Creation: 01. 07. 2015 at 12:28 AM
	Last update: 01. 07. 2015 at 12:28 AM
	Last check: 01. 07. 2015 at 12:28 AM

[Create table](#)



## Flowcharts

I will be using a flowchart diagram to demonstrate the process of using the website. This will be features such as logging in and searching for products. This will be helpful for demonstrating the efficiency of the method I am going to be using for the website. The purpose of a flowchart is to demonstrate how each thing will flow into each other. This can be demonstrated to someone through the use of the flowchart diagram and can show how the login information is used within the website or how when someone searches for a product it loads it from the database. This will also stop errors from being made as a flowchart demonstrates how the system is meant to flow into one another. If any error is made you can then see the issue through the flowchart as it will show what could be causing it to not work properly.

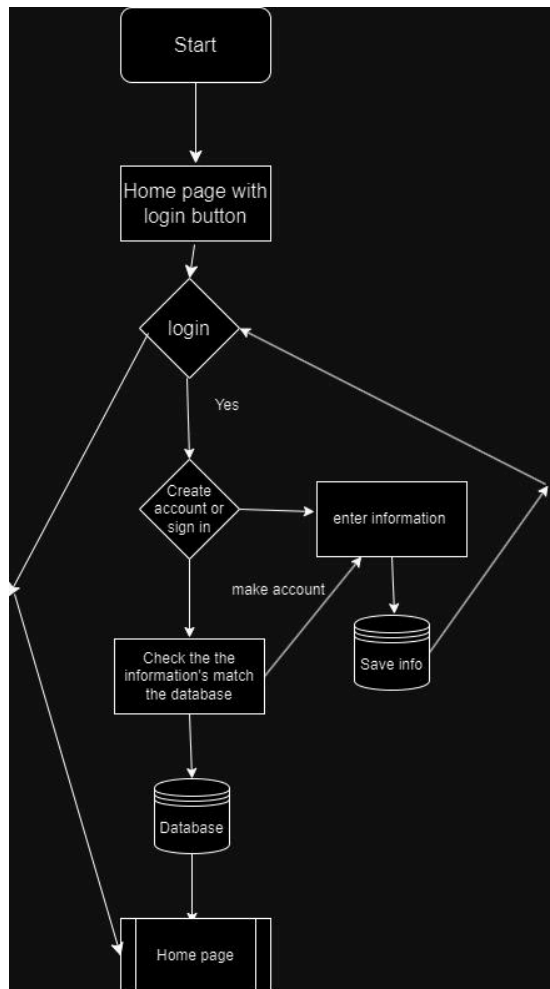
## Client-server architecture

The project will be designed to use the client server side architecture. This is where the client's device, who in this case will be the customers of the website, will access the website through contacting the server to load the pages and products from the database. This method is more appropriate for ecommerce sites as all the products need to be stored into a database where the server file will access the database on the server and send the product the client's device is looking for. This method is more optimal as if I did the project in a client to client architecture it would not work as effectively and you will need to keep updating the website for new products which will require it to go into downtime. This project will also be using HTTPS to make it more secure, and making customer information safe like payment details and personal data which complies with the data protection act 1998.

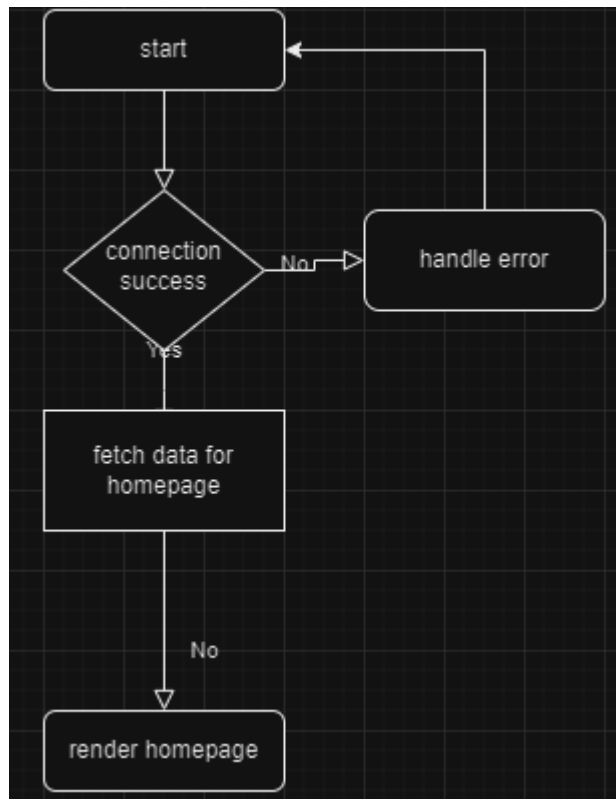
## Login

This flow chart demonstrates the process of logging into the website and how it will work. It shows the system will display the home page with a login button so that if the user wants to, they can sign in to save time inputting information about them when trying to purchase a product. If the login information is found as matches in the database it will allow the user to sign in and reload the home page to show this. If no information is found to do with the account they will be given a screen to make an account for the website if they wish to which will then save their details in the database. This can be seen below:



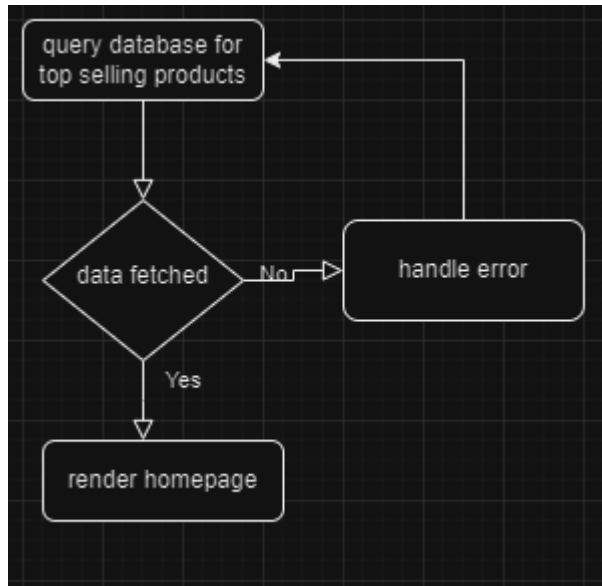


## Handle database connection

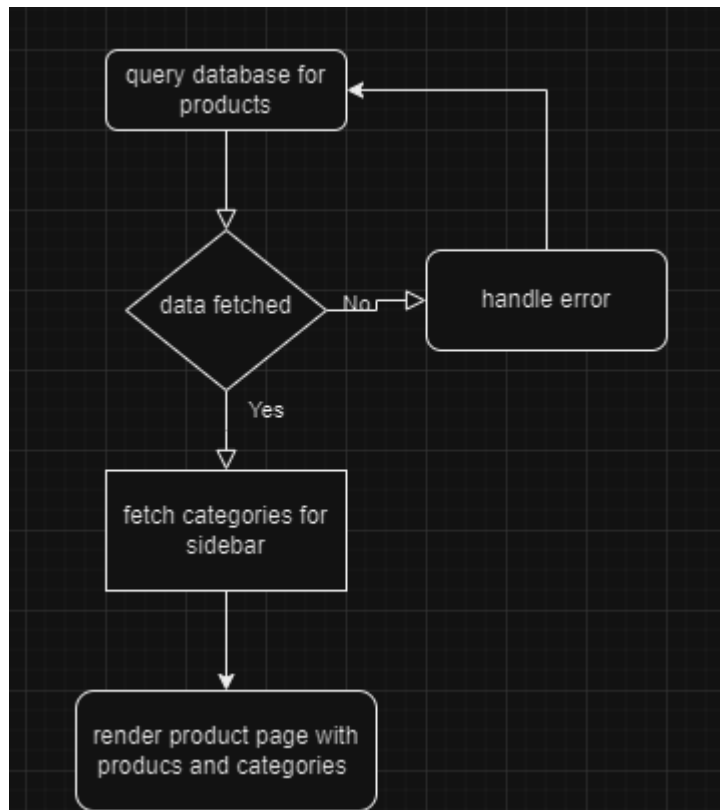


The algorithm I've created manages user login, database interactions, session handling, and routing in Node.js. It checks user credentials, secure authentication, and maintains login status across pages using session management. Database queries retrieve product info and handle shopping cart operations. My algorithm defines routes for rendering forms, displaying products, and processing checkout.

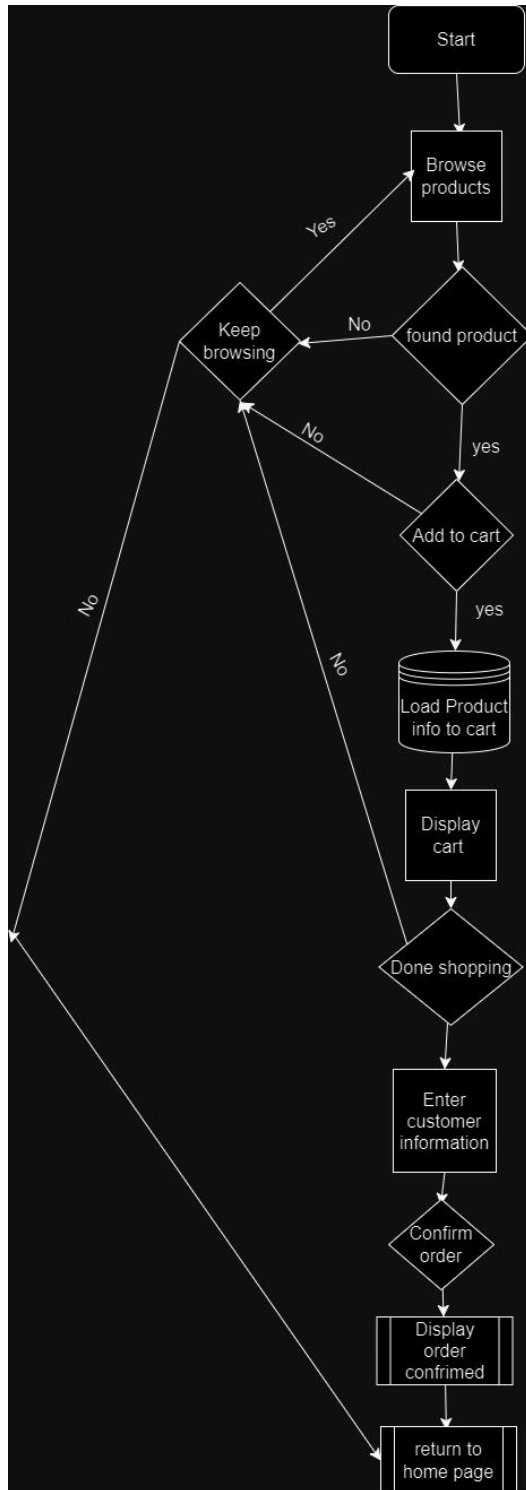
## Fetch data for homepage



## Fetch all products



## Adding product to cart



This flowchart shows the process of adding a product to the cart page and how it will be dynamically loaded. This will be done through the database being made and the product information from the database will be loaded into the cart page with the product name, price and a small image.

This will be an effective method to do this rather than hard coding each part of the page. The method is a common method used by most e-commerce companies and allows for the system to work effectively and be good with load times. This system will work the same way that the top selling products section works and how that is loaded from the database.

## Test Data

The test I will be doing on my website will be with the database and this would be making sure all information is inputted and saved as the correct data type. This will also need to be tested with the images in the website as the method I will be using to load them from the database can have errors occur and this will need to be making the website not format properly. The method I'm going to be using to load the images is using text so that the user inputting products on the system will have to type the image name and the file type for example "4090.png". This is a simpler and easier method of doing this but can cause some human error or the image file not being in the right folder or path. This will be tested through multiple different methods and different file types in case it doesn't work with one.

## Validation

While developing the website for J-Tech I need to ensure that data is accurate and secure across all different systems, data validation plays a key role in this. I'm going to use HTML attributes such as "required" to validate form inputs from the customer, or implement client-side JavaScript checks to provide instant feedback to users. On the server side, it is important to validate the type, length, and format of data to prevent security risks and maintain data integrity, especially when interacting with databases. Using a multi-layered verification approach can significantly improve the overall reliability and user experience of J-techs website.

## Iterative Development Test Data

I will be testing each section of the website after I have fully developed the final product. This will be vital to the project as it will show that it should be error proof.

Website section	Test data	Explanation
Database	I need to verify that the database test the accounts for various different things such as valid data types and all information needed has been inputted.	This is needed as if the database has missing information or certain data not being stored as the correct data types it can cause issues to occur when adding products to the site or when trying to sign in.
Create account	Here I need to check that all required fields of information have been inputted into the system.	This is vital for the website as if one field is left blank it will cause the database to have missing information within it causing the user to not be able to make another account or login.

Login	For this part I will need to check with an existing account that it allows it to login with the correct credentials and the same with what happens when the wrong credentials are entered.	This is needed as if users are not able to login to the system this could cause issues as if they are trying to quickly purchase a product. If this is not done it could cause a loss in potential customers.
-------	--	---

# Implementation

## First iteration: design and structure

In the first iteration I focused on creating its visual layout using HTML and CSS. Starting with key features like the top selling products section and a rotating banner, I ensured the frontend structure was done before moving on to backend tasks. This approach helped me from going back and forward within the development process,

## Second iteration:

In the second iteration, I focused my attention on the backend development of the website. With the frontend layout established in the initial phase, I began implementing the necessary functionalities to make the website fully functional. This included setting up user authentication systems, database interactions, and other backend processes required to support features like user login, product management, and session handling. By focusing on backend development in this iteration, it made it a seamless transition with the frontend layout being already made this ensured the website had it look already made and the functionality done.

## Third iteration: final changes

In the final iteration, I concentrated on refining the website to ensure optimal performance and user experience. This phase involved testing, bug fixes I addressed any remaining issues identified during testing, ensuring that all aspects of the website functioned smoothly. This included fixing any bugs in the frontend and backend, optimising page loading times, and improving overall responsiveness.

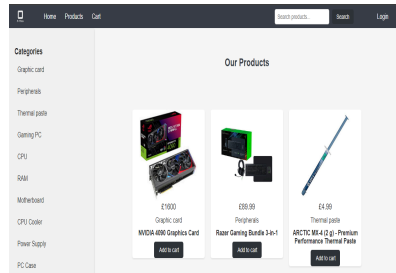
## Test data

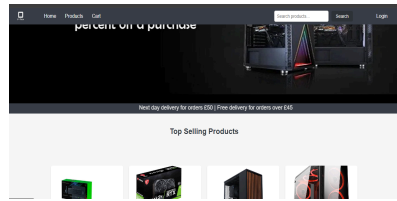
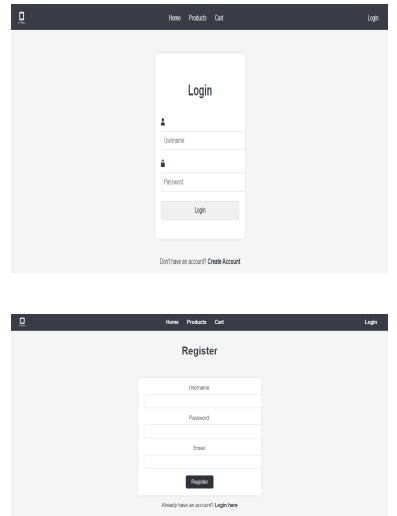
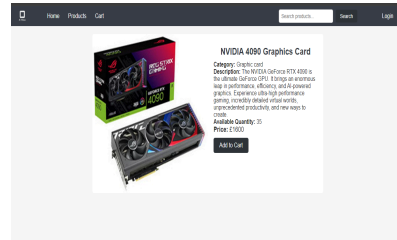
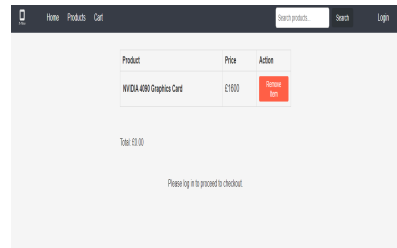
What im testing	Expected output	Actual output
User authentication	Should allow the user to login and if they don't have the credentials it takes them to create an account page.	Let the user sign in and take them to the cart page.
Adding products	Once the product is added to the page it should be there once the site is refreshed	Products appear live and all info is correct as it appears the same as it is in the database
Removing item from cart	The item should disappear from the list	Item disappears from the list



Session management	The session is maintained after logging in The session is closed after logging out	The session stays open when logged in and allows for the user to place their order and it stops them from ordering once they are not signed in.
Database interactions	Products should be fetched and stored on the page with the right information. Items should be added to the cart with the correct info	Products were fetched correctly and they all appeared in the cart with the correct info without any errors.
Route handling	Pages should be fetched and displayed dynamically such as home page product page ect Submitting info for login and checkout should work correctly.	The pages are loaded dynamically and all data info that is needed to be collected is being processed correctly
Middleware functions	The data base should connected automatically and same for user authentication	All functions executed correctly and user authentication worked.

## Test features

What im testing	What it should do	Actual outcome	Evidence
Search bar	Should display the page with all the products that have the words they typed in	The page loads with all products that have words that relate in it	 <p>The screenshot shows a website with a search bar at the top right containing the text 'GPU'. Below the search bar, the page displays 'Our Products' with three items: 'NVIDIA RTX 3080 Graphics Card' for £1099, 'Razer Gaming Mouse 3x-1' for £89.99, and 'ARCTIC M40 40g - Premium Performance Thermal Paste' for £4.99. Each item has an 'Add to cart' button. On the left side of the page, there is a 'Categories' list including Graphic card, Peripherals, Thermal paste, Gaming PC, CPU, SSD, Motherboard, CPU Cooler, Power Supply, and PC Case.</p>

Top selling products	Should display all products with the highest number of sales	Displays certain products with the least amount of stock left	
User registration	Invalid info should make the user make a account Valid input should allow the user to sign in	Users are able to register successfully with valid information, and appropriate error messages are being displayed for invalid inputs.	
Products details	Product information should display when the user tries to access the page.	All information appears in a clean layout	
Shopping cart	Items should appear in the shopping cart Items should disappear from the cart when removed Quantity of products should be able to change	Users are able to add, update, and remove items from the shopping cart seamlessly.	

## First iteration of Solution

### Main Javascript File

The server file which allows for the whole website to run was made with Express.js and MySQL. It handles user logins, database connections, product management, and shopping carts. The setup includes tools for loading the pages, parsing requests, managing sessions, and setting up the database connections. Users can log in securely, and their shopping carts keep track of added products on their account. All products are displayed on the site, and users can search for them by name, category, or type.

```
const express = require('express');

const ejs = require('ejs');

const bodyParser = require('body-parser');

const mysql = require('mysql');

const session = require('express-session');

const path = require('path');

const bcrypt = require('bcrypt');


const app = express();

app.use(express.static(path.join(__dirname, 'public')));


// Login system code

const connectionLogin = mysql.createConnection({

  host: 'localhost',

  user: 'root',

  password: '',

  database: 'nodelogin'

});
```

```
app.use(session({
  secret: 'secret',
  resave: true,
  saveUninitialized: true
}));

app.use(express.json());

app.use(express.urlencoded({ extended: true }));

app.use(express.static(path.join(__dirname, 'static')));

app.get('/login', function (req, res) {
  // Render login template
  res.sendFile(path.join(__dirname + '/views/pages/login.html'));
});

app.post('/auth', function (req, res) {
  // Capture the input fields
  let username = req.body.username;
  let password = req.body.password;

  if (username && password) {
    connectionLogin.query('SELECT * FROM accounts WHERE username = ?', [username], function (error, results, fields) {
      if (error) throw error;
```

```
        if (results.length > 0) {

            // User found, compare passwords using bcrypt

            bcrypt.compare(password, results[0].password, function
(err, bcryptResult) {

                if (err) throw err;

                if (bcryptResult) {

                    req.session.loggedin = true;

                    req.session.username = username;

                    res.redirect('/');

                } else {

                    res.send('Incorrect Username and/or
Password!');

                }

                res.end();

            });

        } else {

            res.send('Incorrect Username and/or Password!');

            res.end();

        }

    });

} else {

    res.send('Please enter both Username and Password!');
```

```
        res.end();

    }

});

app.get('/dashboard', function (req, res) {

    if (req.session.loggedin) {

        res.send('Welcome back, ' + req.session.username + '!');

    } else {

        res.send('Please login to view this page!');

    }

    res.end();

});

// Create a MySQL connection pool

const pool = mysql.createPool({

    host: "localhost",

    user: "root",

    password: "",

    database: "node_j-tech"

});

app.set('view engine', 'ejs');

app.use(bodyParser.urlencoded({ extended: true }));

app.use(session({ secret: "secret" }));
```

```
// Middleware for database connection
app.use((req, res, next) => {
  pool.getConnection((err, connection) => {
    if (err) return next(err);
    req.connection = connection;
    next();
  });
});

// Middleware to release database connection
app.use((req, res, next) => {
  req.connection.release();
  next();
});

// Middleware to calculate total and store in session
app.use((req, res, next) => {
  // Ensure req.session.cart is defined before calculating total
  if (req.session.cart) {
    const total = calculateTotal(req.session.cart);
    req.session.total = total;
    console.log('Total calculated:', total);
  }
});
```

```
        next();
    });

// Function to check if a product is in the cart
function isProductInCart(cart, ID) {
    for (let i = 0; i < cart.length; i++) {
        if (cart[i].id == ID) {
            return true;
        }
    }

    return false;
}

// Function to calculate the total in the cart
function calculateTotal(cart) {
    let total = 0;

    for (let i = 0; i < cart.length; i++) {
        // If we are offering a discounted price
        if (cart[i].Sale_price && !isNaN(cart[i].Sale_price) &&
!isNaN(cart[i].Quantity)) {
            total += cart[i].Sale_price * cart[i].Quantity;
        } else if (!isNaN(cart[i].Price) && !isNaN(cart[i].Quantity)) {
            total += cart[i].Price * cart[i].Quantity;
        }
    }
}
```



```
    }

    }

    return total.toFixed(2); // Convert total to string with two
    decimal places
  }

// Route to fetch top-selling products for the home page
app.get('/', function (req, res, next) {

  req.connection.query("SELECT * FROM products WHERE Quantity <= 25",
  function (err, result) {

    if (err) return next(err);

    res.render('pages/index', { result: result });

  });

});

// Route to fetch all products without quantity filtering
app.get('/products', function (req, res, next) {

  req.connection.query("SELECT * FROM products", function (err,
  result) {

    if (err) return next(err);

    // Fetch categories for the sidebar

    req.connection.query("SELECT DISTINCT Category FROM products",
    function (err, categories) {

      if (err) return next(err);
```

```
        // Render the products page with products and categories
        res.render('pages/products', { result: result, categories:
categories.map(category => category.Category) });

    });

});

});

// Route to add a product to the cart
app.post('/add_to_cart', function (req, res) {

    var ID = req.body.id;

    var Name = req.body.Name;

    var Price = req.body.Price;

    var Sale_price = req.body.Sale_price;

    var Quantity = req.body.Quantity;

    var Image = req.body.Image;

    var Product = { id: ID, Name: Name, Price: Price, Sale_price:
Sale_price, Quantity: Quantity, Image: Image };

    if (req.session.cart) {

        var cart = req.session.cart;

        if (!isProductInCart(cart, ID)) {

            cart.push(Product);
```

```
    }

    } else {

        req.session.cart = [Product];

    }

    // Calculate total

    const total = calculateTotal(req.session.cart);

    req.session.total = total;

    // Return to cart page

    res.redirect('/product/' + ID);

});

// Route to display the cart

app.get('/cart', function (req, res, next) {

    var cart = req.session.cart;

    var total = req.session.total;

    console.log('Total in /cart route:', total);

    // Pass the loggedin variable to the cart template

    res.render('pages/cart', { cart: cart, total: total, loggedin:
req.session.loggedin });

});
```

```
// Route to fetch categories for the home page

app.get('/home', function(req, res, next) {

    req.connection.query("SELECT DISTINCT Category FROM products",
function(err, categories) {

        if (err) return next(err);

        res.render('pages/index', { categories: categories.map(category
=> category.Category), result: [] });

    });

});

app.get('/search', function (req, res, next) {

    var query = req.query.query;

    var category = req.query.category;

    if (!query && !category) {

        res.redirect('/products');

        return;

    }

    var sql;

    var params;
```

```
    if (query && category) {

        sql = "SELECT * FROM products WHERE (Name LIKE ? OR Category
LIKE ? OR Type LIKE ?) AND Category = ?";

        params = [`%${query}%`, `%${query}%`, `%${query}%`, category];

    } else if (query) {

        sql = "SELECT * FROM products WHERE Name LIKE ? OR Category
LIKE ? OR Type LIKE ?";

        params = [`%${query}%`, `%${query}%`, `%${query}%`];

    } else if (category) {

        sql = "SELECT * FROM products WHERE Category = ?";

        params = [category];

    }

    req.connection.query(sql, params, function (err, result) {

        if (err) return next(err);

        res.render('pages/search', { result: result, query: query,
category: category });

    });

});

// Route to display individual product details
app.get('/product/:id', function (req, res, next) {

    var productId = req.params.id;

    var sql = "SELECT * FROM products WHERE ID = ?";

    var params = [productId];
```

```
req.connection.query(sql, params, function (err, result) {

    if (err) return next(err);

    // Check if the product is found

    if (result.length === 0) {

        res.status(404).send('Product not found');

        return;

    }

    // Render the product details page

    res.render('pages/single_product', { product: result[0] });

});

});

app.post('/remove_from_cart', function (req, res) {

    var cart = req.session.cart;

    var productIdToRemove = req.body.id;

    // Find the index of the product to remove

    var indexToRemove = -1;

    for (var i = 0; cart && i < cart.length; i++) {

        if (cart[i].id === productIdToRemove) {

            indexToRemove = i;

            break;

        }

    }

});
```

```
    }

    // Remove the product if found
    if (indexToRemove !== -1) {
        cart.splice(indexToRemove, 1);

        // Recalculate the total
        const total = calculateTotal(cart);

        req.session.total = total;
    }

    res.redirect('/cart'); // Redirect back to the cart page
});

// Route to handle checkout form submission
app.post('/process_checkout', function (req, res) {

    // Extract form data from req.body
    const name = req.body.name;
    const address = req.body.address;

    // Additional form fields go here

    // Process the order, store it in the database, etc.

    // Clear the cart after successful checkout
    req.session.cart = [];
```

```
req.session.total = 0;

// Redirect to a thank you or order confirmation page
res.redirect('/thank_you');
});

// Route to render the checkout page
app.get('/checkout', function (req, res) {
  if (req.session.loggedin) {
    // Render the checkout page if the user is logged in
    res.render('pages/checkout.ejs', { total: req.session.total });
  } else {
    // Redirect to the login page if the user is not logged in
    res.redirect('/login');
  }
});

// Route to render the thank you page
app.get('/thank_you', function (req, res) {
  // Render the thank you page
  res.render('pages/thank_you.ejs');
});
```



```
// Start the server

const PORT = process.env.PORT || 8080; // Use environment variable if
available

app.listen(PORT, function () {

  console.log(`Server is running on port ${PORT}`);

});

// Route to render the registration page

app.get('/register', function (req, res) {

  res.sendFile(path.join(__dirname + '/views/pages/register.html'));

});

// Route to handle registration form submission

app.post('/register', function (req, res) {

  let username = req.body.username;

  let password = req.body.password;

  let email = req.body.email;

  if (username && password && email) {

    // Check if the username is already taken

    connectionLogin.query('SELECT * FROM accounts WHERE username =
?', [username], function (error, results) {

      if (error) throw error;
```

```
        if (results.length > 0) {

            res.send('Username already exists. Please choose a
different username.');
```

```
        } else {

            // Hash the password

            bcrypt.hash(password, 10, function (err, hash) {

                if (err) throw err;

                // Insert the new user into the database with the
hashed password

                connectionLogin.query('INSERT INTO accounts
(username, password, email) VALUES (?, ?, ?)', [username, hash, email],
function (error) {

                    if (error) throw error;

                    res.send('Registration successful! You can now
<a href="/login">login</a>.');

                });

            });

        }

    });

} else {

    res.send('Please enter both username, password, and email.');
```

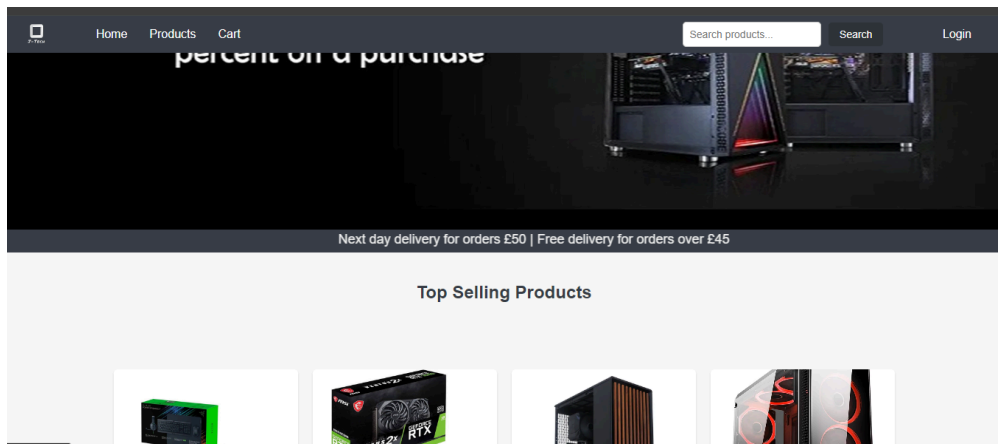
```
    }

});
```

Images of website prototype

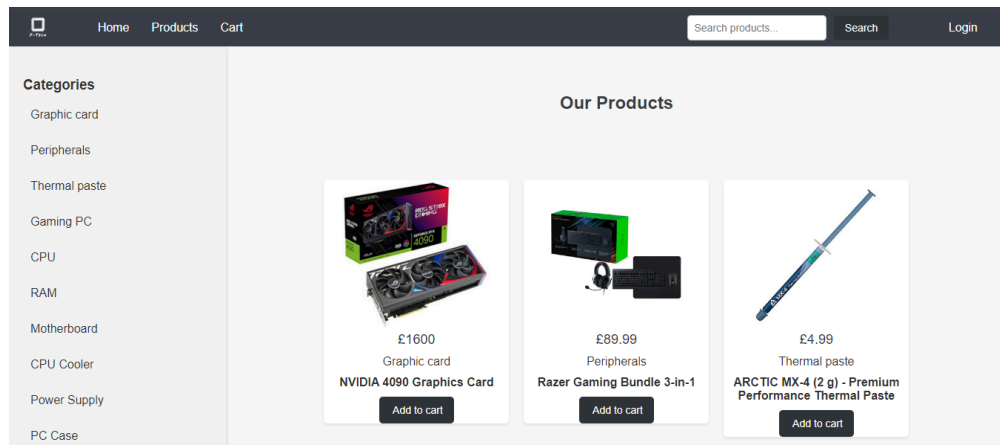
Home page

This image shows the homepage of the website how the site is layed out. I have stuck with the wireframe I made for the website and have now achieved the end product which looks modern and has animations. The nav bar has links to the homepage, product, login and cart.



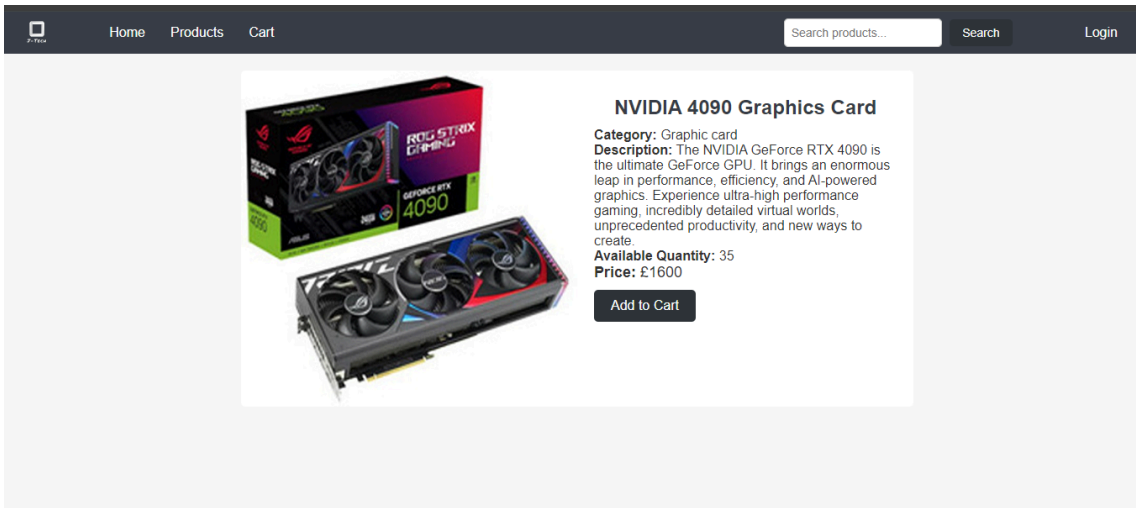
Product page

The product page has the sidebar that shows the type of product that the company sells and this is all dynamic and taken from the server so that if they ever add a new type of product that category will be automatically added to the site. The products are shown in a grid of three and this is the best layout for the site so it is not overloaded and too much for the user. This has the layout I stated I wanted to do in the wireframe and is modern and unique for J-Tech and is functional for customers to purchase products from.



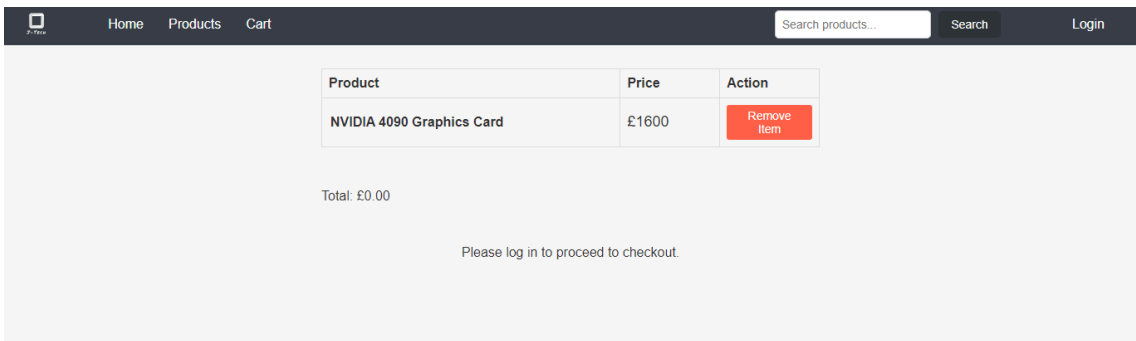
Single product page

This page shows the product description, name, category, price and the quantity this is all loaded from the database. The page follows the generic layout used for most e-commerce websites and is easy for the users to add the product to cart.



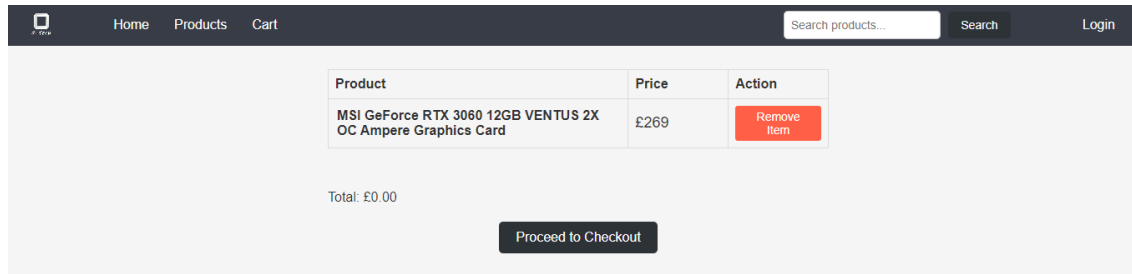
Cart page when not logged in

The cart page changes what is displayed on it whether or not you are signed in. This is a good method to stop users from checking out guests and getting more products per user. The page has a remove item button that deletes the product from the cart page.



## Cart page when logged in

Once the user is logged in for the website the cart page changes what is displayed and the button for checking out is displayed. This button takes the user to the checkout page and this page asks for the users personal information to ship the products to.



The screenshot shows the J-Tech website's cart page. At the top is a dark navigation bar with a logo, links for Home, Products, and Cart, a search bar with the placeholder 'Search products...', and a Login link. The main content area has a light gray background. It features a table with three columns: Product, Price, and Action. The table contains one item: 'MSI GeForce RTX 3060 12GB VENTUS 2X OC Ampere Graphics Card' with a price of '£269'. To the right of the price is a red button labeled 'Remove Item'. Below the table, the text 'Total: £0.00' is displayed. At the bottom center is a dark button labeled 'Proceed to Checkout'.

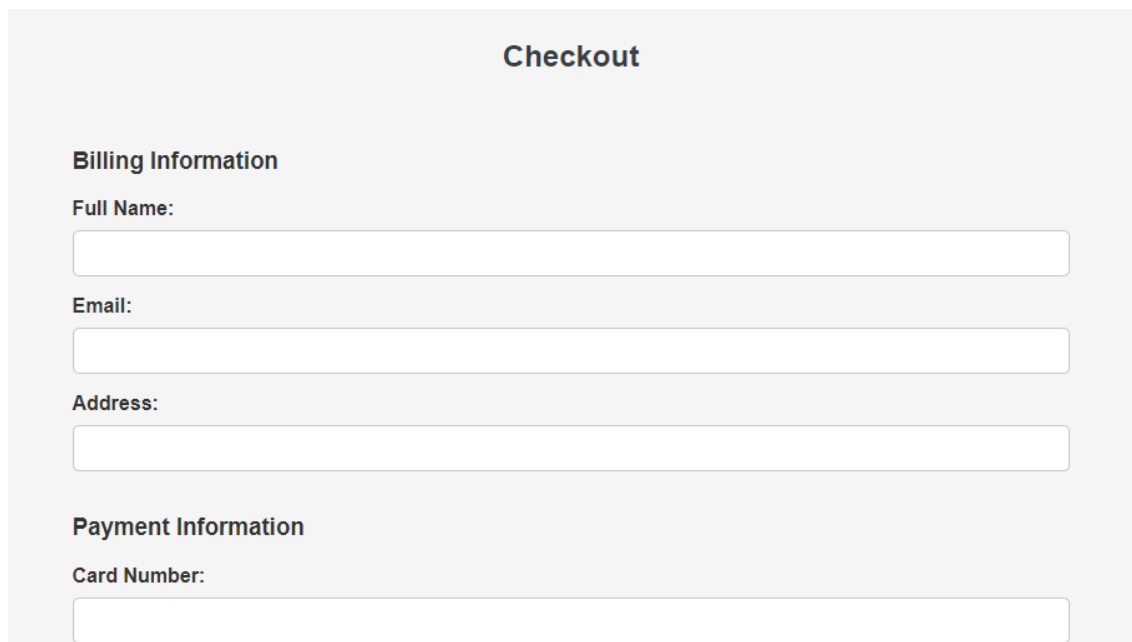
Product	Price	Action
MSI GeForce RTX 3060 12GB VENTUS 2X OC Ampere Graphics Card	£269	<a href="#">Remove Item</a>

Total: £0.00

[Proceed to Checkout](#)

## Checkout page

The checkout page asks for the users information and this will take the information that is inputted and stores it into the orders database. This will later email the information to J-Tech employees so that they do not accidentally ship old orders.



The screenshot shows the J-Tech website's checkout page. The title 'Checkout' is centered at the top. Below it, the section 'Billing Information' is followed by three input fields: 'Full Name:', 'Email:', and 'Address:'. Below these, the section 'Payment Information' is followed by one input field: 'Card Number:'. All input fields are empty and have a light gray border.

### Checkout

**Billing Information**

Full Name:

Email:

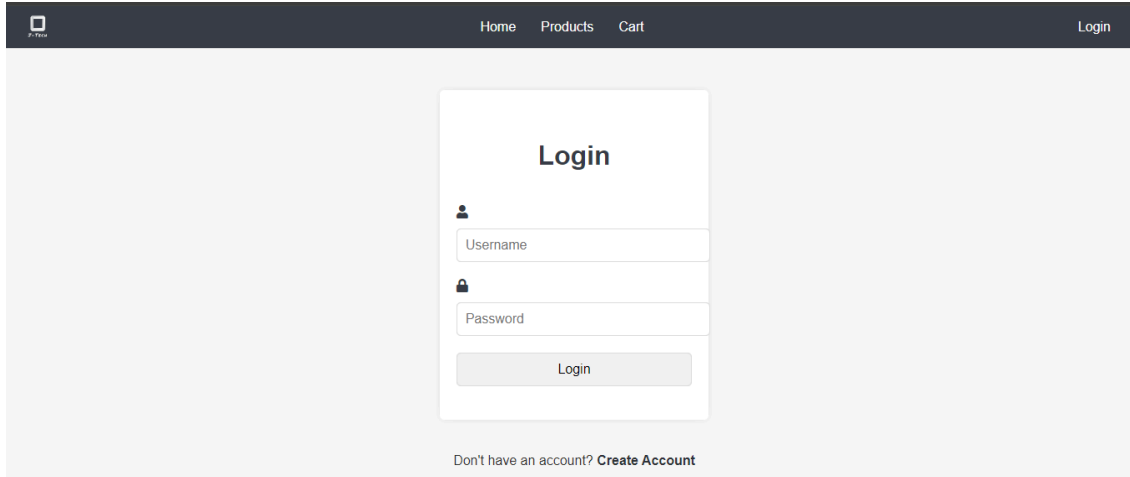
Address:

**Payment Information**

Card Number:

## Login page

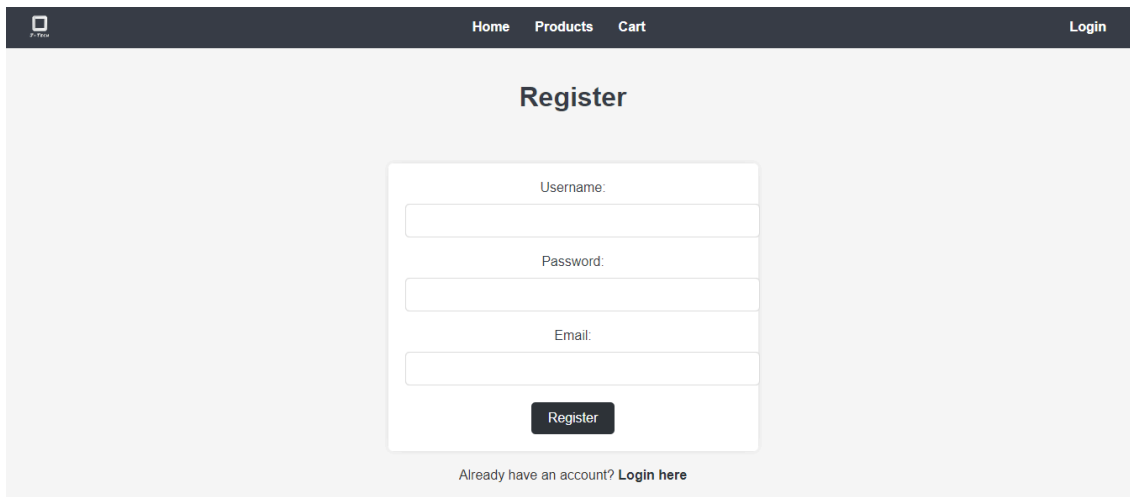
The login page has a simple and generic layout that most websites use and this requests the user to enter their username and password to sign them in. This information is already in the database if they have already made an account. If they do not have an account they can create one by clicking on the create an account page that directs the user to the registration page where they can create an account.



The screenshot shows a web application's login page. At the top, a dark navigation bar contains a logo on the left and links for 'Home', 'Products', 'Cart', and 'Login' on the right. The main content area has a light gray background. In the center, there is a white card titled 'Login'. Inside the card, there is a user icon, a text input field labeled 'Username', a lock icon, a text input field labeled 'Password', and a 'Login' button. Below the card, there is a link that says 'Don't have an account? [Create Account](#)'.

## Register page

The register page has the similar design to the login page but on this side it asks for there email to stop multiple accounts being made. All of this information is stored into the database with the password being encrypted. This page will let you login once the information has been stored in the database and will redirect you to the login page once done.



The screenshot shows a web application's register page. It has the same dark navigation bar as the login page. The main content area has a light gray background. In the center, there is a white card titled 'Register'. Inside the card, there are three text input fields labeled 'Username:', 'Password:', and 'Email:'. Below these fields is a dark 'Register' button. At the bottom of the card, there is a link that says 'Already have an account? [Login here](#)'.

## Evaluation

I have now completed my first prototype of J-Tech Ecommerce website and I need to now evaluate the overall functionality and performance of the website as well as what objectives I have met. My overall opinion on the first prototype of the website is that it is capable of doing what it needs to do and looks good visually. This prototype has met the majority of the objectives that was set apart from one which is the email as I currently do not have the email of the J-Tech employees and how they would like this to be sent to. I believe this prototype is created to the best of my ability using the resources that I had available to me and what was provided to me.

### Objectives

I will outline the objectives that I have met with this prototype of the website. I will also explain how I achieved each objective.

#### Primary objectives

P1 - Customer need to be able to purchase products

P2 - Customers need to be able to search products

P3 - Employees need to be able to add products to the website through a database .

P4 - Need to be able to add product to cart

P5 - Need to be able to total the price in the cart section.

P6 - Need a login system to allow the user to checkout

These are the primary objectives that I was able to complete in this version of the iterative prototype of the website. These are the main objectives for the project as this makes the site functional and work as an ecommerce website without these features the site would not be able to do the task it needs to do. If I did not achieve any of these objectives the website would not work as I need further causing the client to be unhappy as I wouldn't be able to make the product they want. By having these objectives it gave me a clear structure of what I need to do to achieve the goal of having an ecommerce website. These acted as a step by step guide once I had the main page made on what I needed to do next to make the website act as a fully able ecommerce site. This was important as the owners

of J-Tech wanted to get the website up and running ASAP as they wanted to be able to sell products online as soon as they were able to. By them having this website it is a big improvement to the company as it allows for there to be an increase in the number of sales. I personally feel that this is a much greater method to sell their products compared to having physical stores as this is cheaper for them to run and stores only get sales based on the amount of footfall.

### **Maintenance issues**

When adding products there may be issues with the loading of products from the database. Products that are added when a user is using the site will not be displayed to the site unless the page is refreshed by the customer. This causes issues for the user if they don't refresh the page when they are looking for a product that is just added to the site. This is a minor issue for the website but won't cause a large issue for the customers. Another maintenance issue would be if J-Tech wants any changes made to the database this would cause a downtime for the site as if they want to change the fields in it it would cause the site code to have to be changed and database to be modified.

### **Limitations**

One limitation of the current prototype of the website is the absence of not having the employees emails. This causes a loss in the time that the company can be selling products as the employees will have to check the database for the order information and this will slow down the rate they can be shipping products out and fulfilling the orders. By having their phone and access to the email they can then save time on orders as they won't have to search the database for the information it would all be on the email for them.

### **End user Feedback**

I have now gathered feedback to do with the prototype of the site from the people I interviewed earlier in the development of the website about creating the system. I have asked them what they like about the website and as well as what they would like to be changed in the next version or improved to help aid them with adding products to the site.

### **Owners response**

“We love the overall design of the website as it is very clear to what we wanted it to look like visually. This should help with increasing the quantity of sales in the company as customers don't have to go to physical stores to find out. One issue we have with the website is the payment system. We would like to have a paypal, google pay and apple pay plugins to make it easier and quick for users to pay for their products. This is the main change we want done for the website as this will work better for the company when taking payments and is more secure for dealing with payments.”



## **Employees**

"The actual website is functional and works for customers, however the process of adding products is quite difficult to do when you don't know what you are doing. If I was taught properly and given proper training this would be a much easier process. This could be made easier, but I'm not sure how this could be done."

## **Customer opinion (Focus Group hosted by J-Tech Marketing Team)**

This is a review of opinions from a focus group of people and their views and opinions of the website.

"The website design is really good and easier to navigate through"

"The product images are pixelated and have issues with being displayed as they are cut off"

"The website could be more moving as the website feels really still and assets on the site dont move."

"I like the use of the banner to promote products and sales"

"The total doesn't not work and just displays zero. This also is visible in the checkout. I don't know if this is causing issues with payment being received."

## **Usability**

The feedback I have received from the employees, customers and owners of J-Tech have been very useful and informative to the project. All of the feedback they have stated for the website was useful and will help me develop the existing prototype. Some of the issues they addressed in the focus group of customers was a problem with the totaling of the products. Another issue was that employees found it a difficult method to add products to the site and stated they may need some training or another potential method for the adding of products to the system. The term usability has five features: these are if the system is effective, efficient, engaging, error tolerant and easy to learn. I believe I have met around four out of the five of these as there are some errors that are occurring in the current prototype of the website. The overall site is easy and engaging to users visually and pretty straightforward to navigate through. The prototype of the site is effective and efficient in what it needs to do. It works as a fully fledged ecommerce website with some minor issues that can be resolved pretty easily. I personally believe the overall efficiency of the website can be improved with load times this could be done by React js or Next js as this is a method commonly used to increase the optimum load time for websites.

## Errors with the project

Throughout the project the errors that occurred caused issues that needed to be overcome for the website to look visually good and work. One issue that I had to overcome was an issue with the images being dynamically loaded from the database as the path isn't being found as the images were being declared as text and the program would use the text in the database and search for the file in the folder of images. This was fixed by changing the way the algorithm worked as images kept not being displayed correctly. When the database was being coded within the project there was originally no error handling so if data queries didn't work it would not load the page. This was resolved by making error handling for each aspect of the site so if the database couldn't fetch data from the database it would get it keep trying and still load the page without the products until it is fetched. When I originally coded the user input for their username and password it was not hashed and just stored their password not securely and was readable to anyone who has access to the database. When files such as images were being fetched it caused 404 errors to occur when they were not fetched properly which could lead to sensitive information being exposed.

## Future Improvements

The feedback I received from all the individuals I have interviewed will be very helpful and vital to help me make a second iteration of the site as this will act as a set of steps and guidelines to improve the site. One of the improvements I will work on next for the website will be the totaling of the products as one of the customers stated there was an issue when the products total was being displayed in the cart. This is vital for the website to work properly as it is how customers find out the total of multiple products. This could cause issues for the company if they are charging accounts with no money for the products it could cause issues causing a loss of money. Another change that needs to be made to the actual site would be the change of the checkout system. This needs to be changed as the company wants payment plugin buttons to increase the speed users can purchase their products at. There is a plugin that I can use that is available from paypal, apple and google to do this. This would be a simple change to the site that will help the company get the payments they need for the products they are selling. Some minor changes that the site needs would be solving the issue with the product images as some of the images are becoming pixelated or cut off when displayed which can cause the website to look unprofessional. Another change I'm considering making to the site which is a large change that would help the employees when adding products to the website. This could be a potentially different page that is accessible on the website if the user login details have admin privileges that would be cleaner than PHPmyadmin and this would be a more straightforward approach to adding the products to the website as PHPmyadmin can be confusing and difficult to learn when adding products.

The prototype has achieved its main goals, such as letting customers buy products, search for items, add products to the cart, and use a login for checkout. However, one goal about

sending emails to J-Tech employees is not done yet because their email addresses are missing. To solve this, a temporary manual email system could be used until the addresses are available. Some issues were also found such as a delay in showing new products and downtime when changing the database. To improve the site, real-time updates for new products, and Google Pay, and Apple Pay options for payments could be considered

## Conclusion

The first iteration of the website has been successful however there is still room for improvement and some final touches that need to be made. This is the same with most projects as there's always room for improvement. I have met all the primary objectives for the website project. This is an achievement in itself as the objectives help to structure the website. The overall usability of the website is good but not the best that it could be. There are some errors within the site that need to be resolved. To improve the efficiency of the website is a task that might be done later on as it is a big change to do but will make the pages load faster and products so there is no delay.

This iteration of the website has helped me to develop my knowledge of javascript and node js as I had not used much of this language before, only simple bits of it. This has increased my knowledge and if I was asked to make a system like this in the future I would be able to do this with ease and be able to reuse assets from this project and be able to make it faster as I have gained more knowledge on what I'm doing. When i'm next making a website for another company or someone who needs something similar i would be able to do it easier as i have learned from this project thing that went wrong such as what not to do and making sure you have got all the information from the client to make sure you don't have to do multiple changes to the project.

This system will help J-Tech in the future with selling their products to their customers and I will always be available to help them with any issues and improve the site

## Appendix

```
const express = require('express');

const ejs = require('ejs');

const bodyParser = require('body-parser');

const mysql = require('mysql');

const session = require('express-session');

const path = require('path');

const bcrypt = require('bcrypt');


const app = express();

app.use(express.static(path.join(__dirname, 'public')));


// Login system code

const connectionLogin = mysql.createConnection({

  host: 'localhost',

  user: 'root',

  password: '',

  database: 'nodelogin'

});


app.use(session({

  secret: 'secret',

  resave: true,
```

```
        saveUninitialized: true
    }));

app.use(express.json());

app.use(express.urlencoded({ extended: true }));

app.use(express.static(path.join(__dirname, 'static')));

app.get('/login', function (req, res) {

    // Render login template

    res.sendFile(path.join(__dirname + '/views/pages/login.html'));

});

app.post('/auth', function (req, res) {

    // Capture the input fields

    let username = req.body.username;

    let password = req.body.password;

    if (username && password) {

        connectionLogin.query('SELECT * FROM accounts WHERE username = ? ', [username], function (error, results, fields) {

            if (error) throw error;

            if (results.length > 0) {

                // User found, compare passwords using bcrypt
```

```
        bcrypt.compare(password, results[0].password, function
(err, bcryptResult) {

            if (err) throw err;

            if (bcryptResult) {

                req.session.loggedin = true;

                req.session.username = username;

                res.redirect('/');

            } else {

                res.send('Incorrect Username and/or
Password!');

            }

            res.end();

        });

    } else {

        res.send('Incorrect Username and/or Password!');

        res.end();

    }

});

} else {

    res.send('Please enter both Username and Password!');

    res.end();

}

});
```

```
app.get('/dashboard', function (req, res) {

  if (req.session.loggedin) {

    res.send('Welcome back, ' + req.session.username + '!');

  } else {

    res.send('Please login to view this page!');

  }

  res.end();

});

// Create a MySQL connection pool

const pool = mysql.createPool({

  host: "localhost",

  user: "root",

  password: "",

  database: "node_j-tech"

});

app.set('view engine', 'ejs');

app.use(bodyParser.urlencoded({ extended: true }));

app.use(session({ secret: "secret" }));

// Middleware for database connection

app.use((req, res, next) => {
```

```
pool.getConnection((err, connection) => {

  if (err) return next(err);

  req.connection = connection;

  next();

});

});

// Middleware to release database connection
app.use((req, res, next) => {

  req.connection.release();

  next();

});

// Middleware to calculate total and store in session
app.use((req, res, next) => {

  // Ensure req.session.cart is defined before calculating total
  if (req.session.cart) {

    const total = calculateTotal(req.session.cart);

    req.session.total = total;

    console.log('Total calculated:', total);

  }

  next();

});
```



```
// Function to check if a product is in the cart

function isProductInCart(cart, ID) {

    for (let i = 0; i < cart.length; i++) {

        if (cart[i].id == ID) {

            return true;

        }

    }

    return false;

}

// Function to calculate the total in the cart

function calculateTotal(cart) {

    let total = 0;

    for (let i = 0; i < cart.length; i++) {

        // If we are offering a discounted price

        if (cart[i].Sale_price && !isNaN(cart[i].Sale_price) &&
!isNaN(cart[i].Quantity)) {

            total += cart[i].Sale_price * cart[i].Quantity;

        } else if (!isNaN(cart[i].Price) && !isNaN(cart[i].Quantity)) {

            total += cart[i].Price * cart[i].Quantity;

        }

    }

}
```

```
        return total.toFixed(2); // Convert total to string with two
decimal places
    }

    // Route to fetch top-selling products for the home page
    app.get('/', function (req, res, next) {

        req.connection.query("SELECT * FROM products WHERE Quantity <= 25",
function (err, result) {

            if (err) return next(err);

            res.render('pages/index', { result: result });

        });
    });

    // Route to fetch all products without quantity filtering
    app.get('/products', function (req, res, next) {

        req.connection.query("SELECT * FROM products", function (err,
result) {

            if (err) return next(err);

            // Fetch categories for the sidebar

            req.connection.query("SELECT DISTINCT Category FROM products",
function (err, categories) {

                if (err) return next(err);

                // Render the products page with products and categories
```

```
        res.render('pages/products', { result: result, categories:
categories.map(category => category.Category) });

        });

    });

});

// Route to add a product to the cart
app.post('/add_to_cart', function (req, res) {

    var ID = req.body.id;

    var Name = req.body.Name;

    var Price = req.body.Price;

    var Sale_price = req.body.Sale_price;

    var Quantity = req.body.Quantity;

    var Image = req.body.Image;

    var Product = { id: ID, Name: Name, Price: Price, Sale_price:
Sale_price, Quantity: Quantity, Image: Image };

    if (req.session.cart) {

        var cart = req.session.cart;

        if (!isProductInCart(cart, ID)) {

            cart.push(Product);

        }

    } else {
```

```
    req.session.cart = [Product];

  }

  // Calculate total

  const total = calculateTotal(req.session.cart);

  req.session.total = total;

  // Return to cart page

  res.redirect('/product/' + ID);
});

// Route to display the cart
app.get('/cart', function (req, res, next) {

  var cart = req.session.cart;

  var total = req.session.total;

  console.log('Total in /cart route:', total);

  // Pass the loggedin variable to the cart template

  res.render('pages/cart', { cart: cart, total: total, loggedin:
req.session.loggedin });
});
```

```
// Route to fetch categories for the home page

app.get('/home', function(req, res, next) {

    req.connection.query("SELECT DISTINCT Category FROM products",
function(err, categories) {

        if (err) return next(err);

        res.render('pages/index', { categories: categories.map(category
=> category.Category), result: [] });

    });

});

app.get('/search', function (req, res, next) {

    var query = req.query.query;

    var category = req.query.category;

    if (!query && !category) {

        res.redirect('/products');

        return;

    }

    var sql;

    var params;

    if (query && category) {

        sql = "SELECT * FROM products WHERE (Name LIKE ? OR Category
LIKE ? OR Type LIKE ?) AND Category = ?";
```

```
        params = [`%${query}%`, `%${query}%`, `%${query}%`, category];

    } else if (query) {

        sql = "SELECT * FROM products WHERE Name LIKE ? OR Category
LIKE ? OR Type LIKE ?";

        params = [`%${query}%`, `%${query}%`, `%${query}%`];

    } else if (category) {

        sql = "SELECT * FROM products WHERE Category = ?";

        params = [category];

    }

    req.connection.query(sql, params, function (err, result) {

        if (err) return next(err);

        res.render('pages/search', { result: result, query: query,
category: category });

    });

});

// Route to display individual product details
app.get('/product/:id', function (req, res, next) {

    var productId = req.params.id;

    var sql = "SELECT * FROM products WHERE ID = ?";

    var params = [productId];

    req.connection.query(sql, params, function (err, result) {
```

```
        if (err) return next(err);

        // Check if the product is found

        if (result.length === 0) {

            res.status(404).send('Product not found');

            return;

        }

        // Render the product details page

        res.render('pages/single_product', { product: result[0] });

    });

});

app.post('/remove_from_cart', function (req, res) {

    var cart = req.session.cart;

    var productIdToRemove = req.body.id;

    // Find the index of the product to remove

    var indexToRemove = -1;

    for (var i = 0; cart && i < cart.length; i++) {

        if (cart[i].id === productIdToRemove) {

            indexToRemove = i;

            break;

        }

    }

}
```

```
// Remove the product if found

if (indexToRemove !== -1) {

    cart.splice(indexToRemove, 1);

    // Recalculate the total

    const total = calculateTotal(cart);

    req.session.total = total;

}

res.redirect('/cart'); // Redirect back to the cart page
});

// Route to handle checkout form submission
app.post('/process_checkout', function (req, res) {

    // Extract form data from req.body

    const name = req.body.name;

    const address = req.body.address;

    // Additional form fields go here

    // Process the order, store it in the database, etc.

    // Clear the cart after successful checkout

    req.session.cart = [];

    req.session.total = 0;
```



```
    // Redirect to a thank you or order confirmation page

    res.redirect('/thank_you');

  });

// Route to render the checkout page
app.get('/checkout', function (req, res) {

  if (req.session.loggedin) {

    // Render the checkout page if the user is logged in

    res.render('pages/checkout.ejs', { total: req.session.total });

  } else {

    // Redirect to the login page if the user is not logged in

    res.redirect('/login');

  }

});

// Route to render the thank you page
app.get('/thank_you', function (req, res) {

  // Render the thank you page

  res.render('pages/thank_you.ejs');

});

// Start the server
```

```
const PORT = process.env.PORT || 8080; // Use environment variable if
available

app.listen(PORT, function () {

  console.log(`Server is running on port ${PORT}`);

});

// Route to render the registration page
app.get('/register', function (req, res) {

  res.sendFile(path.join(__dirname + '/views/pages/register.html'));

});

// Route to handle registration form submission
app.post('/register', function (req, res) {

  let username = req.body.username;

  let password = req.body.password;

  let email = req.body.email;

  if (username && password && email) {

    // Check if the username is already taken

    connectionLogin.query('SELECT * FROM accounts WHERE username =
?', [username], function (error, results) {

      if (error) throw error;

      if (results.length > 0) {
```

```
        res.send('Username already exists. Please choose a
different username.');
```

```
    } else {

        // Hash the password

        bcrypt.hash(password, 10, function (err, hash) {

            if (err) throw err;

            // Insert the new user into the database with the
hashed password

            connectionLogin.query('INSERT INTO accounts
(username, password, email) VALUES (?, ?, ?)', [username, hash, email],
function (error) {

                if (error) throw error;

                res.send('Registration successful! You can now


```
    }

}\);
```


```

## Home page

For the main page of the website I used an EJS file system with javascript code to dynamically render the pages of the website to the end user. I have used some code in

there to load the products from the database dynamically without me having to hardcode each item to the page. I have used this method as it will be helpful to J-Tech when they want to add products to the page as all they have to do is to add the product to the database and have XAMPP running to do so. The file structure of the website is using a main Javascript file to load the website and create all the functions within the site such as the add to cart function.

```
<!DOCTYPE html>

<html lang="en">

<head>

    <!-- sets the screen scale -->

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width,
initial-scale=1.0">

    <!-- Website Title -->

    <title>J-Tech</title>

    <link rel="stylesheet" href="/css/Style.css">

</head>

<body>

    <nav class="nav">

        <!-- Company logo within the navigation bar-->

        <!-- Navigation Bar -->

        <ul>

            <li><a href="/">Home</a></li>

            <li><a href="/products">Products</a></li>

            <li><a href="/cart">Cart</a></li>
```

```
</ul>

<div class="search-bar">

  <form action="/search" method="get">

    <input type="text" id="searchInput" name="query"
placeholder="Search products...">

    <button type="submit" id="searchButton">Search</button>

  </form>

</div>

<div class="login-link">

  <a href="/login">Login</a>

</div>

</nav>

<script>

  // JavaScript code to rotate the banner images

  document.addEventListener("DOMContentLoaded", function() {

    const bannerImages = document.querySelectorAll("#banner
img");

    let currentIndex = 0;

    function showNextImage() {

      // Hide the current image

      bannerImages[currentIndex].classList.remove("active");
```

```

        // Calculate the index of the next image

        currentIndex = (currentIndex + 1) %
bannerImages.length;

        // Show the next image

bannerImages[currentIndex].classList.add("active");

    }

    // Show the first image initially

    bannerImages[currentIndex].classList.add("active");

    // Switch images every 5 seconds

    setInterval(showNextImage, 5000);

});
</script>

<div id="banner">

    <!-- Banner for GPU to promote new offers-->

    <a href=""></a>

    <!-- Banner for Esport to promote esports PC-->

    <a href=""></a>

```

```

        <!-- Banner for discount -->

        <a href=""></a>

        <!-- Banner for motherboard and CPU special offer.-->

        <a href=""></a>

    </div>

    <div class="scrolling-text">

        <p>Next day delivery for orders £50 | Free delivery for orders
over £45</p>

    </div>

    <div class="top-selling-text">

        <h2>Top Selling Products</h2>

    </div>

    <div class="top-selling-products">

        <% result.forEach(function(item) { %>

            <div class="product">

                <div class="product-price">£<%= item.Price %></div>

                <div class="product-name"><%= item.Name %></div>

                <!-- Add to cart function -->

                <form action="/add_to_cart" method="post">

                    <input type="hidden" name="id" value="<%=
item.ID %>">

```

```

        <input type="hidden" name="Name" value="<%=
item.Name %>">

        <input type="hidden" name="Price" value="<%=
item.Price %>">

        <input type="hidden" name="sale_price"
value="<%= item.Sale_price %>">

        <input type="hidden" name="quantity" value="1">

        <input type="hidden" name="Image" value="<%=
item.Image %>">

        <input type="submit" value=" Add to cart"
class="btn btn-primary">

    </form>

</div>

<% }) %>

</div>

</body>

</html>

```

## Product page

The product page is loading the products from the database and the same with the category section this is taking all the categories in the database and making them be presented on the page to make navigating for a certain product easier. This checks if there are any duplicates and makes sure the same category is not being displayed multiple times. The category section uses the search function that is in the server file and this function loads the search page to display all the products to do with the category. The



products all have an add to cart button and when this is pressed it loads a file called single\_product.ejs and this loads all the products information with an add to cart function that loads the product info to the cart page.

```
<!DOCTYPE html>

<html lang="en">

<head>

  <meta charset="UTF-8">

  <meta name="viewport" content="width=device-width,
initial-scale=1.0">

  <link rel="stylesheet" href="/css/Style.css">

  <title>Home</title>

</head>

<body>

<nav class="nav">

  <ul>

    <li><a href="/">Home</a></li>

    <li><a href="/products">Products</a></li>

    <li><a href="/cart">Cart</a></li>

  </ul>

  <div class="search-bar">

    <form action="/search" method="get">

      <input type="text" id="searchInput" name="query"
placeholder="Search products...">
```

```
        <button type="submit" id="searchButton">Search</button>

    </form>

</div>

<div class="login-link">

    <a href="/login">Login</a>

</div>

</nav>

<div class="container">

    <div id="left-navigation">

        <h3>Categories</h3>

        <ul>

            <% categories.forEach(function(category) { %>

                <li><a href="/search?category=<%= category %>"><%= category
%></a></li>

                <% }); %>

            </ul>

        </div>

        <div id="main-content">

            <div class="top-selling-products-text">

                <h2>Our Products</h2>

            </div>

            <div class="top-selling-products">
```

```

    <% result.forEach(function(item) { %>

        <div class="product">

            <div class="product-price">£<%= item.Price %></div>

            <div class="product-category"><%= item.Category %></div>

            <div class="product-name"><%= item.Name %></div>

            <form action="/add_to_cart" method="post">

                <input type="hidden" name="id" value="<%= item.ID
%>">

                <input type="hidden" name="Name" value="<%= item.Name
%>">

                <input type="hidden" name="Price" value="<%=
item.Price %>">

                <input type="hidden" name="sale_price" value="<%=
item.Sale_price %>">

                <input type="hidden" name="quantity" value="1">

                <input type="hidden" name="Image" value="<%=
item.Image %>">

                <input type="submit" value=" Add to cart" class="btn
btn-primary">

            </form>

        </div>

    <% }) %>

</div>

</div>

</div>

```

```
</body>

</html>
```

## Cart page

The cart page has a function to pull the data from the database once the add to cart button has been pressed it loads the data from the product that has been added to cart and loads the price and product name into the table on the page and loads them from the database. It also calculates the total through the use of the total function and this works by pulling all the info needed from the database such as the name and price and it inserts it into the table to be visible to the user. The cart won't let you checkout unless you are signed in. If you are signed out it will say please sign in this acts a step of authentication.

```
<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width,
initial-scale=1.0">

    <link rel="stylesheet" href="/css/Style.css">

    <link rel="stylesheet" href="/css/cart.css">

    <title>Shopping Cart</title>

</head>

<body>

    <nav class="nav">

        <!-- Company logo within the navigation bar-->

        <!-- Navigation Bar -->
```

```
<ul>

  <li><a href="/">Home</a></li>

  <li><a href="/products">Products</a></li>

  <li><a href="/cart">Cart</a></li>

</ul>

<div class="search-bar">

  <form action="/search" method="get">

    <input type="text" id="searchInput" name="query"
placeholder="Search products...">

    <button type="submit" id="searchButton">Search</button>

  </form>

</div>

<div class="login-link">

  <a href="/login">Login</a>

</div>

</nav>

<div class="cart-container">

  <h2>Shopping Cart</h2>

  <div class="table-container">

    <% if (cart && cart.length > 0) { %>

      <table>

        <thead>
```

```

        <tr>

            <th>Product</th>

            <th>Price</th>

            <th>Action</th>

        </tr>

    </thead>

    <tbody>

        <% cart.forEach(function(item) { %>

            <tr>

                <td class="product-name"><%= item.Name
%></td>

                <td class="product-price">£<%=
item.Price %></td>

                <td>

                    <form action="/remove_from_cart"
method="post">

                        <input type="hidden" name="id"
value="<%= item.id %>">

                        <button type="submit"
class="remove-item-btn">Remove Item</button>

                    </form>

                </td>

            </tr>

        <% }); %>

    </tbody>

```

```
        </table>

        <p>Total: £<%= total %></p>

    <% } else { %>

        <p>Your shopping cart is empty.</p>

    <% } %>

</div>

<div class="checkout-button">

    <% if (loggedin) { %>

        <form action="/checkout" method="get">

            <button type="submit" class="checkout-btn">Proceed
to Checkout</button>

        </form>

    <% } else { %>

        <p>Please log in to proceed to checkout.</p>

    <% } %>

</div>

</div>

</div>

</body>

</html>
```

## Single product page

This is the page that the user sees when they click on a product they want to view. This page is created for each product and this gets all the information for the product when the button is pressed to see more. It loads the name, description, quantity, price, category and

the type. This is the best method for J-Tech as when they wish to add more products to the site they do not have to code each product page and all they have to do on their end is load the database through XAMPP and add all the product information that is needed on the database.

```
<!-- product_details.ejs -->

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width,
initial-scale=1.0">

    <link rel="stylesheet" href="/css/Style.css">

    <link rel="stylesheet" href="/css/productStyle.css">

    <title>Product Details</title>

</head>

<body>

    <nav class="nav">

        <ul>

            <li><a href="/">Home</a></li>

            <li><a href="/products">Products</a></li>

            <li><a href="/cart">Cart</a></li>

        </ul>

        <div class="search-bar">

            <form action="/search" method="get">
```



```

        <input type="text" id="searchInput" name="query"
placeholder="Search products...">

        <button type="submit" id="searchButton">Search</button>

    </form>

</div>

<div class="login-link">

    <a href="/login">Login</a>

</div>

</nav>

<div class="product-details-container">

    <div class="product-details">

        <div class="product-info">

            <h2 class="product-name"><%= product.Name %></h2>

            <div
class="product-category"><strong>Category:</strong> <%=
product.Category %></div>

            <div
class="product-description"><strong>Description:</strong> <%=
product.Description %></div>

            <div class="product-quantity"><strong>Available
Quantity:</strong> <%= product.Quantity %></div>

            <div class="product-price"><strong>Price:</strong> £<%=
product.Price %></div>

            <form action="/add_to_cart" method="post">

```

```

        <input type="hidden" name="id" value="<%=
product.ID %>">

        <input type="hidden" name="Name" value="<%=
product.Name %>">

        <input type="hidden" name="Price" value="<%=
product.Price %>">

        <input type="hidden" name="Sale_price" value="<%=
product.Sale_price %>">

        <input type="hidden" name="quantity" value="1">

        <input type="hidden" name="Image" value="<%=
product.Image %>">

        <button type="submit" class="add-to-cart-btn">Add
to Cart</button>

    </form>

</div>

</div>

</div>

</body>

</html>

```

## Search page

The search page works mainly on a function that is in the server file. This function uses an SQL statement to look for the products that have been searched for in the search bar which is accessible throughout the site. This search bar uses the sql statement that looks for any products that have that word within it and this pulls all the products up and displays them on this page.

```

<!DOCTYPE html>

<html lang="en">

```

```
<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width,
initial-scale=1.0">

    <link rel="stylesheet" href="/css/Style.css">

    <title>Search Results</title>

</head>

<body>

    <nav class="nav">

        <ul>

            <li><a href="/">Home</a></li>

            <li><a href="/products">Products</a></li>

            <li><a href="/cart">Cart</a></li>

        </ul>

        <div class="search-bar">

            <form action="/search" method="get">

                <input type="text" id="searchInput" name="query"
placeholder="Search products...">

                <button type="submit" id="searchButton">Search</button>

            </form>

        </div>

        <div class="login-link">
```

```

        <a href="/login">Login</a>

    </div>

</nav>

<h2>Search Results for "<%= query %>"</h2>

<% result.forEach(function(item) { %>

    <div class="product">

        <div class="product-price">&lt;%= item.Price %></div>

        <div class="product-category"><%= item.Category %></div>

        <div class="product-name"><%= item.Name %></div>

        <form action="/add_to_cart" method="post">

            <input type="hidden" name="id" value="<%= item.ID %>">

            <input type="hidden" name="Name" value="<%= item.Name
%>">

            <input type="hidden" name="Price" value="<%= item.Price
%>">

            <input type="hidden" name="sale_price" value="<%=
item.Sale_price %>">

            <input type="hidden" name="quantity" value="1">

            <input type="hidden" name="Image" value="<%= item.Image
%>">

            <input type="submit" value=" Add to cart" class="btn
btn-primary">

        </form>

    </div>

```

```
    <% }) %>

</body>

</html>
```

## Login page

The login page has a generic login layout within it and requests the user for the username and password to sign them in. This searches the login databases to find the user's name and password. This is encrypted in the database to stop people breaching the data protection act. This is encrypted through a command that is in the server file and this saves the password as a hashed string. The page also has a create account button underneath the main box that directs the user to the register page.

```
<!DOCTYPE html>

<html>

<head>

    <meta charset="utf-8">

    <meta name="viewport" content="width=device-width,
minimum-scale=1">

    <title>Login</title>

    <link rel="stylesheet"
href="https://use.fontawesome.com/releases/v5.7.1/css/all.css">

    <link href="/css/login.css" rel="stylesheet" type="text/css">

    <link href="/css/Style.css" rel="stylesheet" type="text/css">

</head>

<body>

    <nav class="nav">

        <!-- Company logo within the navigation bar-->

        
```

```

    <!-- Navigation Bar -->

    <ul>

        <li><a href="/">Home</a></li>

        <li><a href="/products">Products</a></li>

        <li><a href="/cart">Cart</a></li>

    </ul>

    <div class="login-link">

        <a href="/login">Login</a>

    </div>

</nav>

<div class="login-container">

    <div class="login">

        <h1>Login</h1>

        <form action="/auth" method="post">

            <label for="username">

                <i class="fas fa-user"></i>

            </label>

            <input type="text" name="username"
placeholder="Username" id="username" required>

            <label for="password">

                <i class="fas fa-lock"></i>

            </label>

            <input type="password" name="password"
placeholder="Password" id="password" required>

```

```
        <input type="submit" value="Login">

    </form>

</div>

<div class="create-account">

    <p>Don't have an account? <a href="/register">Create
Account</a></p>

</div>

</div>

</body>

</html>
```

## Register page

The register page asks the user for their username, they want password and email. This checks that their username does not already exist in the database and if it does it displays a message that it is not allowed. There is a function within the server file that takes all the information that has been inputted and takes it into the database once that has been done. It loads a page that says you can now successfully login now and this page takes the user straight to the login page and allows them to login.

```
<!DOCTYPE html>

<html>

<head>

    <meta charset="utf-8">

    <meta name="viewport" content="width=device-width,
minimum-scale=1">

    <title>Login</title>

    <link rel="stylesheet"
href="https://use.fontawesome.com/releases/v5.7.1/css/all.css">
```

```
<link href="/css/login.css" rel="stylesheet" type="text/css">

<link href="/css/Style.css" rel="stylesheet" type="text/css">

</head>

<body>

  <nav class="nav">

    <!-- Company logo within the navigation bar-->

    <!-- Navigation Bar -->

    <ul>

      <li><a href="/">Home</a></li>

      <li><a href="/products">Products</a></li>

      <li><a href="/cart">Cart</a></li>

    </ul>

    <div class="login-link">

      <a href="/login">Login</a>

    </div>

  </nav>

  <div class="login-container">

    <div class="login">

      <h1>Login</h1>

      <form action="/auth" method="post">

        <label for="username">

          <i class="fas fa-user"></i>

        </label>
```



```
        <input type="text" name="username"
placeholder="Username" id="username" required>

        <label for="password">

            <i class="fas fa-lock"></i>

        </label>

        <input type="password" name="password"
placeholder="Password" id="password" required>

        <input type="submit" value="Login">

    </form>

</div>

<div class="create-account">

    <p>Don't have an account? <a href="/register">Create
Account</a></p>

</div>

</div>

</body>

</html>
```

## Checkout page

The checkout page uses a function to display the total that is already being displayed in the cart page. The checkout page displays the product total and asks for the user information which is stored into a orders database which later down the line I will make a function for that emails J-Tech the products they have purchased and where to deliver it to

```
. <!DOCTYPE html>
```

```
<html lang="en">
```

```
<head>
```

```
<meta charset="UTF-8">

<meta name="viewport" content="width=device-width,
initial-scale=1.0">

<link rel="stylesheet" href="/css/Style.css">

<link rel="stylesheet" href="/css/checkout.css"> <!-- Add a new CSS
file for checkout styles -->

<title>Checkout</title>
</head>
<body>

  <nav class="nav">

    <!-- Company logo within the navigation bar-->

    <!-- Navigation Bar -->

    <ul>

      <li><a href="/">Home</a></li>

      <li><a href="/products">Products</a></li>

      <li><a href="/cart">Cart</a></li>

    </ul>

    <div class="login-link">

      <a href="/login">Login</a>

    </div>

  </nav>
```

```
<div class="checkout-container">

  <h2>Checkout</h2>

  <form action="/complete_order" method="post">

    <!-- Billing Information -->

    <h3>Billing Information</h3>

    <label for="fullname">Full Name:</label>

    <input type="text" id="fullname" name="fullname" required>

    <label for="email">Email:</label>

    <input type="email" id="email" name="email" required>

    <label for="address">Address:</label>

    <input type="text" id="address" name="address" required>

    <!-- Payment Information -->

    <h3>Payment Information</h3>

    <label for="cardnumber">Card Number:</label>

    <input type="text" id="cardnumber" name="cardnumber"
required>

    <label for="expdate">Expiration Date:</label>

    <input type="text" id="expdate" name="expdate"
placeholder="MM/YYYY" required>
```

```
<label for="cvv">CVV:</label>

<input type="text" id="cvv" name="cvv" required>

<!-- Order Summary (You may need to fetch this from the
server) -->

<h3>Order Summary</h3>

<!-- Display the products in the cart and their prices -->

<!-- Total Price (You may need to fetch this from the
server) -->

<p>Total: £<%= total %></p>

<!-- Submit Button -->

<button type="submit">Complete Order</button>

</form>

</div>

</body>

</html>
```

## Order Confirmation

The order confirmation page loads the information about the order that the user has placed and says the total charge and the items they have bought are all loaded from the database.

```
<!DOCTYPE html>

<html lang="en">

<head>
```

```
<meta charset="UTF-8">

<meta name="viewport" content="width=device-width,
initial-scale=1.0">

<link rel="stylesheet" href="/css/Style.css">

<title>Order Confirmation</title>
</head>
<body>

  <nav class="nav">

    <!-- Company logo within the navigation bar-->

    <!-- Navigation Bar -->

    <ul>

      <li><a href="/">Home</a></li>

      <li><a href="/products">Products</a></li>

      <li><a href="/cart">Cart</a></li>

    </ul>

    <div class="login-link">

      <a href="/login">Login</a>

    </div>

  </nav>

  <div class="confirmation-container">

    <h2>Order Confirmation</h2>
```

```
<p>Thank you for your order! Your purchase is being
processed.</p>

<p>Order Details:</p>

<ul>

  <% orderItems.forEach(function(item) { %>

    <li><strong><%= item.Name %></strong> - Quantity: <%=
item.Quantity %>, Price: £<%= item.Price %></li>

    <% }); %>

  </ul>

<p>Total Amount: £<%= total %></p>

<p>Your order has been confirmed. An email with the order
details will be sent to your registered email address.</p>

</div>

</body>

</html>
```

## Script file

This script file has some functions that are used within the website. This file is mainly used for the moving and interactive elements in the site. The search bar and rotating banner can be seen in the file to help make the site more interactive and animated.

```
// Search Bar

document.addEventListener('DOMContentLoaded', function () {

  const searchInput = document.getElementById('searchInput');

  const searchButton = document.getElementById('searchButton');

  const products = document.querySelectorAll('.product');
```

```
searchButton.addEventListener('click', function () {

    const searchTerm = searchInput.value.toLowerCase();

    products.forEach(function (product) {

        const productName =
product.querySelector('.product-name').innerText.toLowerCase();

        const productCategory =
product.querySelector('.product-category').innerText.toLowerCase();

        if (productName.includes(searchTerm) ||
productCategory.includes(searchTerm)) {

            product.style.display = 'block';

        } else {

            product.style.display = 'none';

        }

    });

});

});

});

//code to rotate the banner images

document.addEventListener("DOMContentLoaded", function() {

    const bannerImages = document.querySelectorAll("#banner img");

    let currentIndex = 0;
```

```
function showNextImage() {  
    // Hide the current image  
    bannerImages[currentImageIndex].classList.remove("active");  
  
    // Calculate the index of the next image  
    currentImageIndex = (currentImageIndex + 1) %  
bannerImages.length;  
  
    // Show the next image  
    bannerImages[currentImageIndex].classList.add("active");  
}  
  
// Show the first image initially  
bannerImages[currentImageIndex].classList.add("active");  
  
// Switch images every 5 seconds  
setInterval(showNextImage, 5000);  
});
```

## CSS files

I have used multiple different css files for the site. This is for each of the pages such as cart, checkout, login, product page and the register page. There is a main style file which is used for the whole site and this is the largest of all as it sets up the nav bar, search bar, image size, text size, background colour, text colour and any animations.



## Cart css

```
/* Reset default margin and padding */

body, h1, h2, nav, ul, li {

    margin: 0;

    padding: 0;

    box-sizing: border-box;

}


/* Apply styles to the navigation bar */

nav {

    position: fixed;

    top: 0;

    left: 0;

    right: 0;

    background-color: #393E46;

    height: 50px;

    display: flex;

    align-items: center;

    padding: 10px 20px;

    z-index: 100;

}


nav img {
```

```
    height: 40px;
  }

nav ul {
    list-style-type: none;
    display: flex;
}

nav li {
    margin-left: 10px;
}

nav li a {
    color: #fff;
    text-decoration: none;
    font-size: 16px;
    padding: 10px;
}

body {
    font-family: 'Arial', sans-serif;
    background-color: #f9f9f9;
    color: #333;
}
```

```
.CartContainer {  
  
  margin: 20px auto; /* Center the container */  
  
  margin-top: 20px;  
  
  background-color: #fff;  
  
  padding: 20px;  
  
  border-radius: 8px;  
  
  box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);  
  
  max-width: 600px; /* Limit the width of the container */  
}  
  
.Cart-Item {  
  
  display: flex;  
  
  align-items: center;  
  
  justify-content: space-between;  
  
  border-bottom: 1px solid #ddd;  
  
  padding: 10px 0;  
}  
  
.Cart-Item img {  
  
  max-width: 60px;  
  
  max-height: 60px;  
  
  margin-right: 10px;  
}
```

```
.Item-Details {  
  flex-grow: 1;  
}  
  
.Item-Details h3 {  
  margin: 0;  
  font-size: 18px;  
}  
  
.Item-Details p {  
  margin: 5px 0;  
  color: #777;  
}  
  
.remove-item-btn {  
  background-color: #FF6347;  
  color: #fff;  
  border: none;  
  padding: 5px 10px;  
  border-radius: 3px;  
  cursor: pointer;  
  font-size: 14px;  
}
```

```
.table-container {  
  
    margin: 0 auto;  
  
    max-width: 600px;  
  
}  
  
table {  
  
    width: 100%;  
  
    border-collapse: collapse;  
  
    margin-top: 50px;  
  
}  
  
th, td {  
  
    border: 1px solid #ddd;  
  
    padding: 8px;  
  
    text-align: left;  
  
}  
  
.product-name {  
  
    width: 60%;  
  
}  
  
.product-price {  
  
    width: 20%;
```

```
}

.remove-item-btn {

  background-color: #FF6347;

  color: #fff;

  border: none;

  padding: 5px 10px;

  border-radius: 3px;

  cursor: pointer;

  font-size: 14px;

}

p{

  margin-top: 50px;

}

.checkout-button {

  margin-top: 20px;

  text-align: center;

}

.checkout-btn {

  background-color: #2d363b;

  color: #fff;

  border: none;

  padding: 10px 20px;
```

```
border-radius: 5px;

cursor: pointer;

font-size: 16px;

transition: background-color 0.3s ease;
}

.checkout-btn:hover {

background-color: #1a2124;

}
```

## Checkout css

```
/* Styling for the checkout container */

.checkout-container {

max-width: 800px;

margin: 0 auto;

padding: 20px;

}

/* Styling for form elements */

form {

display: flex;

flex-direction: column;

}
```

```
label {  
  
    margin-bottom: 8px;  
  
    font-weight: bold;  
  
}  
  
input {  
  
    padding: 10px;  
  
    margin-bottom: 15px;  
  
    border: 1px solid #ccc;  
  
    border-radius: 5px;  
  
}  
  
button {  
  
    background-color: #2d363b;  
  
    color: #fff;  
  
    border: none;  
  
    border-radius: 5px;  
  
    cursor: pointer;  
  
    font-size: 16px;  
  
    padding: 10px 15px;  
  
}  
  
button:hover {
```



```
background-color: #222;  
  
}
```

## Login css

```
body {  
  
    font-family: Arial, sans-serif;  
  
    background-color: #F9F9F9;  
  
    color: #333;  
  
    padding: 50px 20px;  
  
    text-align: center;  
  
}  
  
nav {  
  
    position: fixed;  
  
    top: 0;  
  
    left: 0;  
  
    right: 0;  
  
    background-color: #393E46;  
  
    height: 50px;  
  
    display: flex;  
  
    align-items: center;  
  
    padding: 0 20px;
```

```
    z-index: 100;
}

nav img {
    height: 40px;
}

nav ul {
    list-style-type: none;
    display: flex;
}

nav li {
    margin-left: 10px;
}

nav li a {
    color: #fff;
    text-decoration: none;
    font-size: 16px;
    padding: 10px;
}

.login-container {
```

```
    display: flex;

    flex-direction: column;

    align-items: center;

    justify-content: center;

    margin-top: 50px;
}

.login {

    max-width: 400px;

    margin-bottom: 20px;

    background-color: #fff;

    border-radius: 5px;

    box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);

    padding: 20px;
}

h1 {

    color: #393E46;
}

form {

    text-align: left;
}
```

```
label {  
  
    display: block;  
  
    margin-bottom: 10px;  
  
    color: #393E46;  
  
}  
  
input {  
  
    width: 100%;  
  
    padding: 10px;  
  
    margin-bottom: 20px;  
  
    border: 1px solid #ddd;  
  
    border-radius: 5px;  
  
    font-size: 16px;  
  
}  
  
button {  
  
    background-color: #2d363b;  
  
    color: #fff;  
  
    border: none;  
  
    border-radius: 5px;  
  
    cursor: pointer;  
  
    font-size: 16px;  
  
    padding: 10px 20px;  
  
}
```

```
button:hover {  
    background-color: #222;  
}  
  
.create-account p {  
    margin-top: 20px;  
}  
  
.create-account a {  
    color: #2d363b;  
    text-decoration: none;  
    font-weight: bold;  
}  
  
.create-account a:hover {  
    text-decoration: underline;  
}
```

Product css

---

```
/* Container for product details */

.product-details-container {

    display: flex;

    justify-content: center;

    align-items: center;

    padding: 20px;

}

/* Individual product details */

.product-details {

    display: flex;

    flex-wrap: wrap;

    max-width: 800px;

}

.product-image {

    flex: 0 0 50%;

    max-width: 50%;

    max-height: 100%; /* Allow the image to take up the full height of
its container */

    border-radius: 5px 0 0 5px;

}

.product-info {
```

```
    flex: 0 0 50%;

    max-width: 50%;

    box-sizing: border-box;

    padding: 20px;

    background-color: #fff;

    border-radius: 0 5px 5px 0;
}

/* Adjustments for responsiveness */
@media (max-width: 768px) {

    .product-details {

        flex-direction: column; /* Stack items vertically on small screens
    */

    }

    .product-image,

    .product-info {

        max-width: 100%;

        flex: 0 0 100%;

    }

}

.add-to-cart-btn {

    background-color: #2d363b;
```

```
    color: #fff;

    padding: 10px 20px;

    border: none;

    border-radius: 5px;

    cursor: pointer;

    font-size: 16px;

    transition: background-color 0.3s ease;
}

.add-to-cart-btn:hover {

    background-color: #222;
}
```

## Register css

```
body {

    font-family: Arial, sans-serif;

    background-color: #F9F9F9;

    color: #333;

    padding: 50px 20px;

    text-align: center;
}
```



```
h1 {  
  
    color: #393E46;  
  
}  
  
form {  
  
    max-width: 400px;  
  
    margin: 20px auto;  
  
    padding: 20px;  
  
    background-color: #fff;  
  
    border-radius: 5px;  
  
    box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);  
  
}  
  
label {  
  
    display: block;  
  
    margin-bottom: 10px;  
  
    color: #393E46;  
  
}  
  
input {  
  
    width: 100%;  
  
    padding: 10px;  
  
    margin-bottom: 20px;  
  
    border: 1px solid #ddd;
```

```
border-radius: 5px;

font-size: 16px;
}

button {

background-color: #2d363b;

color: #fff;

border: none;

border-radius: 5px;

cursor: pointer;

font-size: 16px;

padding: 10px 20px;
}

button:hover {

background-color: #222;
}

p {

margin-top: 20px;
}

a {

color: #2d363b;
```

```
text-decoration: none;

font-weight: bold;

}
```

## Main CSS file

```
/* Reset default margin and padding */

body, h1, h2, nav, ul, li {

  margin: 0;

  padding: 0;

  box-sizing: border-box;

}

nav {

  position: fixed;

  top: 0;

  left: 0;

  right: 0;

  background-color: #393E46;

  height: 50px;

  align-items: center;

  padding: 0 20px;

  z-index: 100;

  display: flex;

  justify-content: space-between;
```

```
}

nav img {

    height: 40px;

}

nav ul {

    list-style-type: none;

    display: flex;

}

nav li {

    margin-left: 10px;

}

nav li a {

    color: #fff;

    text-decoration: none;

    font-size: 16px;

    padding: 10px;

}

nav li:hover {

    background-color: #222;
```



```
h1, h2 {  
  
    text-align: center;  
  
    padding: 40px 0;  
  
    font-family: Arial, sans-serif;  
  
    color: #393E46;  
  
}  
  
body {  
  
    font-family: Arial, sans-serif;  
  
    background-color: #F9F9F9;  
  
    color: #333;  
  
    padding-top: 50px;  
  
}  
  
/* styles the banner to fit the screen */  
  
#banner {  
  
    background-color: #f2f2f2;  
  
    display: flex;  
  
    justify-content: center;  
  
    align-items: flex-start;  
  
    height: 300px;  
  
    width: 100vw;  
  
    margin-left: calc(-50vw + 50%);  
}
```

```
margin-right: calc(-50vw + 50%);

z-index: 2;

position: relative;
}

#banner img {

  max-width: 100%;

  height: auto;

  margin: 0;

  cursor: pointer;

  display: none;
}

#banner img.active {

  display: block;
}

.top-selling-products {

  display: flex;

  justify-content: center;

  align-items: flex-start;

  flex-wrap: wrap;

  padding: 20px;
```

```
margin-top: 20px;

position: relative;

z-index: 1;
}

.top-selling-products::after {

  content: "";

  display: table;

  clear: both;
}

.product {

  width: 250px;

  margin: 10px;

  text-align: center;

  background-color: #fff;

  border-radius: 5px;

  box-shadow: 0 2px 5px rgba(0, 0, 0, 0.1);
}

.product img {

  width: 200px;

  height: 200px;

  object-fit: cover;
```



```
border-radius: 5px 5px 0 0;
}

.product-name {
  font-weight: bold;
  padding: 10px;
}

.product-price {
  font-size: 18px;
  color: #333;
  padding-bottom: 10px;
}

.product-button {
  display: inline-block;
  background-color: #393E46;
  color: #fff;
  text-decoration: none;
  padding: 10px 20px;
  border-radius: 5px;
  transition: background-color 0.3s ease;
}
```

```
.product:hover {  
    transform: translateY(-5px);  
}  
  
.product-button:hover {  
    background-color: #222;  
}  
  
/* Styles for the scrolling text container */  
.scrolling-text {  
    width: 100%;  
    height: 10px;  
    position: relative;  
    background-color: #393E46;  
    margin-top: 150px;  
    padding: 13px 0;  
    z-index: 1;  
}  
  
/* Styles for the individual text element */  
.scrolling-text p {  
    white-space: nowrap;  
    position: absolute;  
    width: max-content;
```

```
height: 100%;

margin: 0;

line-height: 10px;

color: #f2f2f2;

font-size: 18px;

animation: scrollText 12s linear infinite;
}

/* Keyframes animation for scrolling the text */
@keyframes scrollText {

  0% {

    transform: translateX(100%);

  }

  100% {

    transform: translateX(-100%);

  }

}

.btn-primary {

  background-color: #2d363b;

  color: #fff;

  padding: 10px 15px;

  border: none;

  border-radius: 5px;
}
```

```
    cursor: pointer;

    font-size: 14px;
}

.container {

    display: flex;

    margin-top: 2px;
}

#left-navigation {

    width: 20%;

    background-color: #f2f2f2;

    padding: 20px;

    box-shadow: 2px 0 5px rgba(0, 0, 0, 0.1);

    z-index: 99;
}

#main-content {

    width: 80%;

    padding: 20px;
}

#left-navigation h3 {
```

```
    color: #333;

    margin-bottom: 10px;
}

#left-navigation ul {

    list-style-type: none;

    padding: 0;
}

#left-navigation a {

    display: block;

    color: #333;

    text-decoration: none;

    padding: 10px;

    margin-bottom: 10px;

    transition: background-color 0.3s ease;
}

#left-navigation a:hover {

    background-color: #ddd;
}

.search-bar {

    align-items: right;
```

```
margin-left: 550px;

margin-right: 20px; /* Adjust the margin as needed */

}

#searchInput {

padding: 8px;

border: 1px solid #ccc;

border-radius: 5px;

margin-right: 5px;

font-size: 14px;

}

#searchButton {

background-color: #2d363b;

color: #fff;

border: none;

border-radius: 5px;

cursor: pointer;

font-size: 14px;

padding: 8px 15px;

}
```