

ONLINE COMPUTER HARDWARE STORE MANAGEMENT

A PROJECT COMPONENT REPORT

Submitted by

DOMNIC SAM J

(Reg. No. 202004032)

ABISHEK B

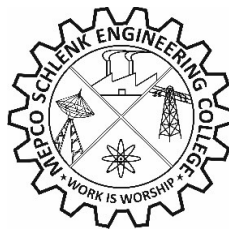
(Reg. No. 202004252)

*for the Theory Cum Project Component
of*

19CS694 – WEB USER INTERFACE DESIGN

during

VI Semester – 2022 – 2023



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

MEPCO SCHLENK ENGINEERING COLLEGE, SIVAKASI

(An Autonomous Institution affiliated to Anna University Chennai)

April 2023

MEPCO SCHLENK ENGINEERING COLLEGE, SIVAKASI

(An Autonomous Institution affiliated to Anna University Chennai)

Department of Computer Science and Engineering

BONAFIDE CERTIFICATE

Certified that this project component report titled **ONLINE COMPUTER
HARDWARE STORE MANAGEMENT** is the bonafide work of **J. DOMNIC SAM
(Reg. No. 202004032), B. ABISHEK (Reg. No. 202004252)** who carried out this work
under my guidance for the Theory cum Project Component course “**19CS694 – WEB
USER INTERFACE DESIGN**” during the sixth semester.

Dr. M. S. BHUVANESWARI, M.E., Ph.D.
Associate Professor
Course Instructor
Department of Computer Science & Engg.
Mepco Schlenk Engineering College
Sivakasi.

Dr. J. RAJA SEKAR, M.E., Ph.D.
Professor
Head of the Department
Department of Computer Science & Engg.
Mepco Schlenk Engineering College
Sivakasi.

Submitted for Viva-Voice Examination held at **MEPCO SCHLENK ENGINEERING
COLLEGE (Autonomous), SIVAKASI** on/05/2023

Internal Examiner

External Examiner

ABSTRACT

Our online computer hardware store is your one-stop-shop for all your computer hardware needs. Whether you're a gamer, a graphic designer, or simply looking to upgrade your home computer, we have a wide range of products to suit your needs. Our store offers the latest and greatest hardware components, including processors, motherboards, graphics cards, RAM, storage devices, and more. We also have a range of peripherals such as keyboards, mice, and monitors to complete your setup. We stock products from all the major brands, including Intel, AMD, NVIDIA, and ASUS, so you can be sure you're getting top-quality components. Our friendly and knowledgeable staff are on hand to answer any questions you may have and help you find the perfect product for your needs. With our fast and reliable shipping, you can have your new hardware delivered right to your door in no time. Plus, our competitive prices mean you can get the latest and greatest hardware without breaking the bank. Shop with confidence at our online computer hardware store and take your computing experience to the next level.

ACKNOWLEDGEMENT

First and foremost, we thank the **LORD ALMIGHTY** for his abundant blessings that is showered upon our past, present and future successful endeavors.

We extend our sincere gratitude to our college management and Principal **Dr. S. Arivazhagan M.E., Ph.D.**, for providing sufficient working environment such as systems and library facilities. We also thank him very much for providing us with adequate lab facilities, which enable us to complete our project.

We would like to extend our heartfelt gratitude to **Dr. J. Raja Sekar M.E., Ph.D.**, Professor and Head, Department of Computer Science and Engineering, Mepco Schlenk Engineering College for giving me the golden opportunity to undertake a project of this nature and for his most valuable guidance given at every phase of our work.

We would also like to extend our gratitude and sincere thanks to **Dr. M. S. Bhuvaneswari M.E., Ph.D.**, Associate Professor, Department of Computer Science and Engineering, Mepco Schlenk Engineering College for being our Project Mentor. She has put her valuable experience and expertise in directing, suggesting and supporting us throughout the Project to bring out the best.

Our sincere thanks to our revered **faculty members and lab technicians** for their help over this project work.

Last but not least, we extend our indebtedness towards our beloved family and our friends for their support which made the project a successful one.

TABLE OF CONTENTS

CHAPTER NO.	TITLE	PAGE NO.
	ABSTRACT	ii
	LIST OF TABLES	v
	LIST OF FIGURES	vi
	LIST OF ABBREVIATIONS	vii
1	INTRODUCTION	1
2	REQUIREMENTS DESCRIPTION	3
	2.1 Functional Requirements	3
	2.2 Non-Functional Requirements	3
3	SYSTEM DESIGN	5
	3.1 Architectural design	5
	3.2 Design Components	7
	3.3 Database Description	7
	3.4 Low Level design	9
	3.5 User Interface design	13
4	SYSTEM IMPLEMENTATION	23
5	RESULTS AND DISCUSSION	26
	CONCLUSION AND FUTURE	
6	ENHANCEMENT(S)	34
APPENDIX –A	SYSTEM REQUIREMENTS	25
APPENDIX –B	SOURCE CODE	26
	REFERENCES	56

LIST OF TABLES

Table No.	Table Caption	Page No.
3.1	Signup Description	7
3.2	Login Description	8
3.3	Product Description	8
3.4	Order Description	8
3.5	Cart Description	9
3.6	Login Details	9
3.7	Signup Details	10
3.8	Add product Details	10
3.9	Add to Cart Details	11
3.10	Place Order Details	11
3.11	Edit Product Details	12
3.12	Delete Product Details	12
5.1	Positive Test Case and result for Login	20
5.2	Negative Test Case and result for Login	20
5.3	Positive Test Case and result for Signup	21
5.4	Negative Test Case and result for Signup	21
5.5	Positive Test Case and result for Add product	21
5.6	Negative Test Case and result for Add product	22
5.7	Positive Test Case and result for Add to Cart	22
5.8	Negative Test Case and result for Add to Cart	23

LIST OF FIGURES

Figure No.	Figure Caption	Page No.
3.1	Architecture Diagram of Online Hardware Store	6
3.2	Home Page of Online Hardware Store	13
3.3	Login Page for Users	13
3.4	Signup Page for Users	14
3.5	Product Details Page	14
3.6	Cart Page	15
3.7	FAQ Page	15
3.8	Admin page	16
3.9	Add Product Page	16
3.10	Update Product Page	17
3.11	Delete Product Page	17

LIST OF ABBREVIATIONS

ABBREVIATION	DESCRIPTION
API	Application Programming Interface
JS	JavaScript
TS	TypeScript
UI	User Interface
RAM	Random Access Memory
HDD	Hard Disk Drive
SDD	Solid Disk Drive
HTML	Hyper Text Markup Language
CSS	Cascading Style sheet

CHAPTER 1

INTRODUCTION

1.1 PERSPECTIVE

Our online computer hardware store is your one-stop-shop for all your computer hardware needs. Whether you're a gamer, a graphic designer, or simply looking to upgrade your home computer, we have a wide range of products to suit your needs. Our store offers the latest and greatest hardware components, including processors, motherboards, graphics cards, RAM, storage devices, and more. We also have a range of peripherals such as keyboards, mice, and monitors to complete your setup. We stock products from all the major brands, including Intel, AMD, NVIDIA, and ASUS, so you can be sure you're getting top-quality components. Our friendly and knowledgeable system is on always available to answer any questions you may have and help you find the perfect product for your needs. Plus, our competitive prices mean you can get the latest and greatest hardware without breaking the bank. Shop with confidence at our online computer hardware store and take your computing experience to the next level.

1.2 OBJECTIVES

The objective of the online computer hardware store project is to create a user-friendly and convenient platform for customers to purchase high-quality hardware products online. The project aims to offer a wide range of hardware products, including RAM, Cooling Fan, AMD and INTEL Processor and other accessories, while also providing a personalized shopping experience. The project will utilize data analytics to offer personalized recommendations based on the customer's previous purchases and browsing history. To ensure a safe shopping experience, the project aims to offer secure payment gateways and provide excellent customer service, including prompt responses to customer queries and timely resolution of issues. By offering high-quality products, personalized recommendations, and excellent customer service, the project aims to increase brand awareness and loyalty while generating revenue for the business. Ultimately, the objective of the online hardware store project is to provide customers with a seamless and hassle-free shopping experience, increasing customer satisfaction and loyalty.

1.3 SCOPE

The scope of the computer hardware store project is to provide customers with a convenient and personalized way to purchase hardware products online. The project aims to offer a wide range of hardware components, including ram, hard disks, GPU , and other accessories, to cater to the diverse needs of customers. Customers will be able to browse and filter products based on various parameters such as price, brand and component types. Moreover, the online computer hardware store to ensure secure payment gateways to protect customer's sensitive information and provide a safe shopping experience. The project will also provide excellent customer service, including prompt responses to customer queries and timely resolution of issues, to ensure customer satisfaction. The scope of the project also includes generating revenue for the business by increasing brand awareness and loyalty through high-quality products, personalized recommendations, and excellent customer service. Overall, the computer hardware store project aims to revolutionize the hardware industry by providing customers with a seamless and hassle-free shopping experience, increasing customer satisfaction, and loyalty.

CHAPTER 2

REQUIREMENT DESCRIPTION

2.1 FUNCTIONAL REQUIREMENTS

The functional requirements of the Online Computer Hardware Store are the collective information about the core operations that are available in the system.

- The system should be able to manage user accounts, including registration, login, and profile management.
- The system should be able to manage product details, such as product images, descriptions, specifications, and pricing. It should also allow the administrator to add, delete or modify products.
- The system should allow customers to add products to their shopping cart, view the total cost of their order, and proceed to checkout. The checkout process should include options for payment and shipping, as well as order confirmation.
- The system should provide search and filter options to help customers find products that meet their specific requirements.
- The system should ensure secure payment gateways to protect customer's sensitive information.
- The system should offer excellent customer service, including prompt responses to customer queries and timely resolution of issues.

2.2 NON-FUNCTIONAL REQUIREMENTS

Non-functional requirements are the quality attributes of the Online Computer Hardware Store that describe how well it performs its intended functions rather than what functions it performs.

- The system should be user-friendly and easy to navigate, providing a smooth shopping experience for customers.
- The system should be able to handle a large volume of traffic and transactions without any downtime or lag.
- The system should ensure data security by implementing measures such as secure payment gateways, data encryption, and firewall protection.

- The system should be compatible with different devices and platforms, such as desktops, laptops, tablets, and mobile phones.
- The system should be scalable to accommodate future growth in traffic and sales.
- The system should be reliable, ensuring that all transactions and customer data are securely processed and stored.
- The system should be available 24/7 to customers, providing them with access to the website at all times.
- The system should be accessible to all customers, including those with disabilities, through features such as screen readers and alternative text.
- The system should comply with relevant laws and regulations, such as data protection and consumer protection laws.

CHAPTER 3

SYSTEM DESIGN

3.1 ARCHITECTURE DESIGN

Architectural diagram implies the flow of the system. The flow starts with the user visiting the homepage of the website. The user can view the various kind of product that are available. The user can also view the detailed description of each hardware product. The user can add the hardware component to the cart that he/she wishes to buy. Once the product is added to the cart, the user can also change the quantity of the hardware product that he/she wishes to buy, that is it can be incremented or decremented. If the user is already logged in, the user proceeds to checkout page. Else, the user is taken to the login page to login to the A2D Online Computer Hardware Store website. If the user is a new user, then the user is taken to the registration page for creating a new account. Then, the user proceeds to check out and enters the shipping address of the products. Then the user chooses the desired payment method, and the order summary is shown to the user.

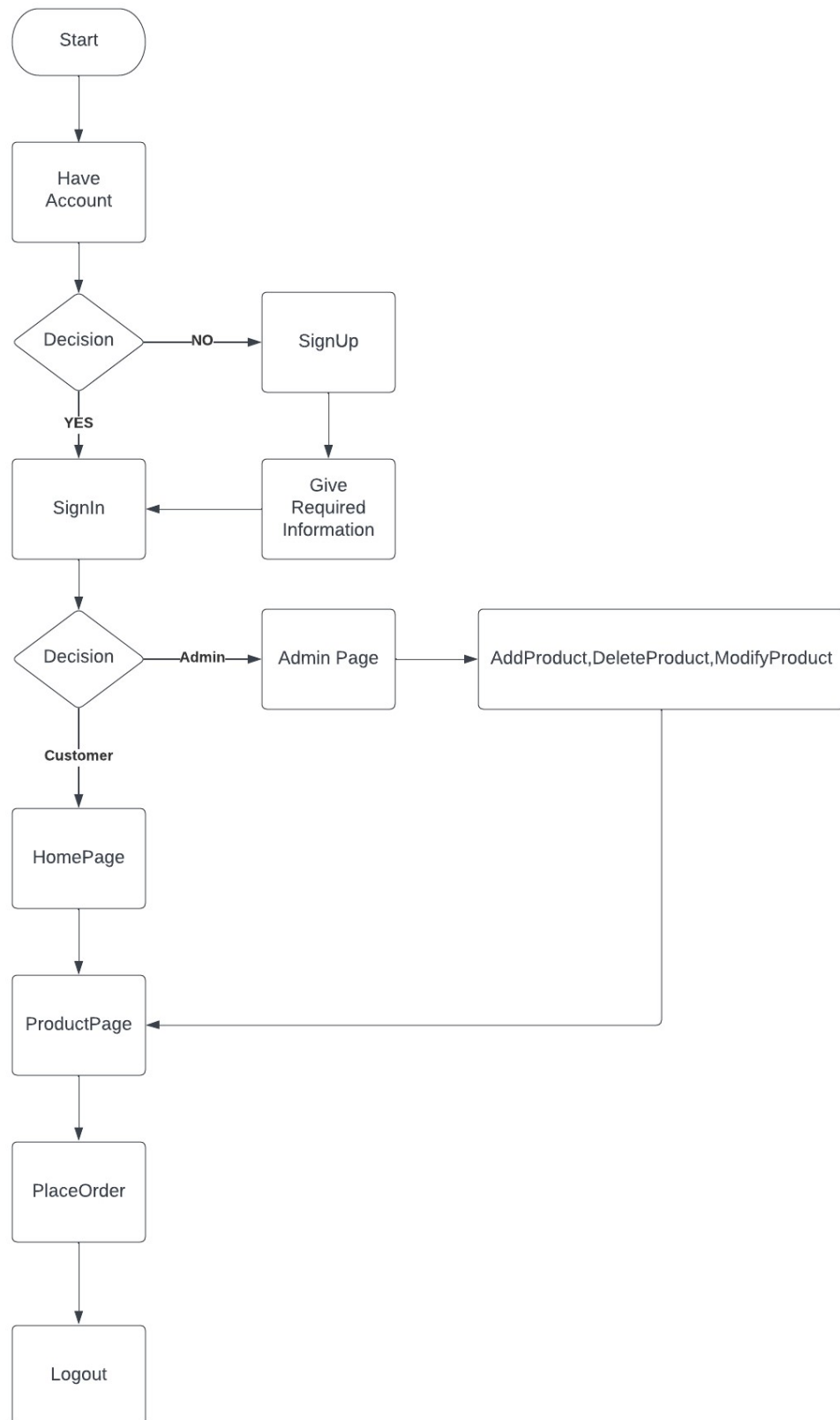


Figure 3.1: Architecture Diagram of Online Computer Hardware Store

3.2 DESIGN COMPONENTS

3.2.1 Front End:

The front end of the Online Computer Hardware Store is developed using the Angular JS framework and Angular Material UI Component Library.

3.2.2 Back End:

Express JS is used to build the backend server for the Online Computer Hardware Store and MongoDB is used as the database in the back end.

3.3 DATABASE DESCRIPTION

Listed below gives a description of database document schemas used for Online Computer Hardware Store

3.3.1 Signup Description

The details about the users stored in the User table are shown in **Table 3.1**.A signup page is a page on your website where users can sign up to use your product.

Table 3.1: User Signup

Attribute Name	Type	Constraint(s)	Description
Username	String	Must be unique	Name of the user
Password	String	Not Null	Password which the user uses to login to the website
Confirm Password	String	Same As Password	Confirm Password check the user enter password is right

3.3.2 Login Description

The details about the User data stored in the login table are shown in **Table 3.2** Shows the login details of the application. It performs the login functionality of the online hardware store. It checks for the credentials whether it is valid or not.

Table 3.2:

Attribute Name	Type	Constraint(s)	Description
Username	String	Must be unique	Name of The Customer
Password	String	Not Null	Customer Password

3.3.3 Product Description

The details about the product data stored in the Product table are shown in **Table 3.3**. A product detail page is a web page on an online hardware store website that provides information on a specific product. This information includes size, color, price, shipping information, reviews, and other relevant information customers want to know before purchasing.

Table 3.3:

Attribute Name	Type	Constraint(s)	Description
ProductName	String	Not Null	Name of The Product
ProductDescription	String	Not Null	Details of the Product
ProductPrice	String	NotLess than 0	Price of the Product
ProductQty	Integer	NotLess than 0	Product available Qty

3.3.4 Order Description

The details about the orders stored in the Order table are shown in **Table 3.4**.

Table 3.4: Order Description

Attribute Name	Type	Constraint(s)	Description
ProductName	String	Not Null	Name of The Product
ProductQty	String	NotLess than 0	Quantity of the Product
TotalAmount	String	NotLess than 0	Total Amount of the Order available in cart
OrderId	String	Must be unique	Order_Id of the placed order

3.3.5 Cart Description

The details about the Products stored in the Cart table are shown in **Table 3.5** provides the cart page of the website. A cart page is an essential part of an e-commerce website.

Table 3.5: Cart Description

Attribute Name	Type	Constraint(s)	Description
ProductName	String	Not Null	Name of The Product
ProductQty	String	NotLess than 0	Quantity of the Product
TotalAmount	String	NotLess than 0	Total Amount of the Order available in cart

3.4 LOW LEVEL DESIGN

The following section illustrates the functionalities of the system. This includes login to the website, register to the website, adding products to the inventory, adding products to the cart, making payment, and placing the order.

3.4.1 Login

Table 3.6 shows the login details of the application.

Table 3.6 Login Details

Files used	login.component.html, login.component.ts, auth.service.ts, user.routes.ts
Short Description	Allows the user to login to the online hardware store website
Arguments	Password
Return	Success/Failure in login
Pre-Condition	The user must have an account in the website
Post-Condition	The home page of the online hardware store will be displayed
Exception	Invalid password
Actors	Admin, User

3.4.2 Signup

Table 3.7 shows the Signup details of the application.

A signup page is a page on your website where users can sign up to use your product. They're designed to capture the email addresses of visitors and make it easy for them to access your product. Usually, a signup page contains a form (a signup form), where users can register by providing necessary information, such as their Username and Password.

Table 3.7 Signup Details

Files used	register.component.html, register.component.ts, user.service.ts, user.routes.ts
Short Description	Allows the user to create a new account in the online hardware store website
Arguments	Name, Password, Confirm Password
Return	Success/Failure in registration
Pre-Condition	The user must not have an account with the e-mail id
Post-Condition	The home page of the online hardware store will be displayed
Exception	Invalid Weak password
Actors	User

3.4.3 Add Product

Table 3.8 shows the add product details of the application.

Table 3.8 Add Product Details

Files used	admin-product-edit.component.html, admin-product-edit.component.ts, product.service.ts, product.routes.ts, admin.guard.ts
Short Description	Allows the admin to add hardware product to the online hardware store website
Arguments	Name, Price, Path to Image, Brand, Category, Description
Return	Success/Failure in adding new hardware product
Pre-Condition	The admin must be logged in to the website
Post-Condition	The hardware inventory page of the online hardware store will be displayed
Exception	Duplicate slug or Invalid path to image
Actors	Admin

3.4.4 Add to Cart

Table 3.9 shows the add to cart details of the application.

Table 3.9 Add to Cart Details

Files used	cart.component.html, cart.component.ts, cart.service.ts, product.routes.ts
Short Description	Allows the user add hardware product to the cart in the online hardware store website
Arguments	Product name, quantity
Return	Success/Failure adding product to cart
Pre-Condition	The user must be logged in to the website
Post-Condition	The cart page of the online hardware store will be displayed
Exception	Product out of stock
Actors	User

3.4.5 Place Order

Table 3.10 shows the place order details of the application.

Table 3.10 Check Out Details

Files used	Checkoutcomponent.html,checkout.component.ts, order.routes.ts, cart.service.ts, order.service.ts
Short Description	Allows the user to login to the online hardware store website
Arguments	product name, Quantity
Return	Success/Failure in Checkout
Pre-Condition	The user must be logged in to the website
Post-Condition	The order summary page of the online hardware store will be displayed
Exception	Invalid shipping address or Product out of stock
Actors	User

3.4.6 Edit Product Details

Table 3.11 shows the edit product details of the application.

Table 3.11 Edit Product Details

Files used	edit.component.html, edit.component.ts,
-------------------	---

	product.service.ts, product.routes.ts, admin.guard.ts
Short Description	Allows the admin to edit hardware product to the online hardware store website
Arguments	Name, Price, Path to Image, Brand, Category, Description
Return	Success/Failure in editing hardware product
Pre-Condition	The admin must be logged in to the website
Post-Condition	The hardware inventory page of the online hardware store will be displayed
Exception	Duplicate slug or Invalid path to image
Actors	Admin

3.4.7 Delete Product Details

Table 3.12 shows the edit product details of the application.

Table 3.12 Delete Product Details

Files used	delete.component.html, delete.component.ts, product.service.ts, product.routes.ts, admin.guard.ts
Short Description	Allows the admin to delete hardware product to the online hardware store website
Arguments	Name, Price, Category, Description
Return	Success/Failure in delete hardware product
Pre-Condition	The admin must be logged in to the website
Post-Condition	The hardware inventory page of the online hardware store will be displayed
Exception	Duplicate slug or Invalid path to image
Actors	Admin

3.5 USER INTERFACE DESIGN

3.5.1 Home Page

Figure 3.2 provides the interface for home page of the website. The homepage of an online hardware store website is very important as it is often the first impression a brand gets to make and usually the first page to tackle in an ecommerce design. It can include promotions, branded hardware products and featured products or categories. It establishes credibility and tells potential customers what your website sells and showcases crucial Calls-to-Action. A good Online hardware store website should always entice visitors to stay and explore your product.

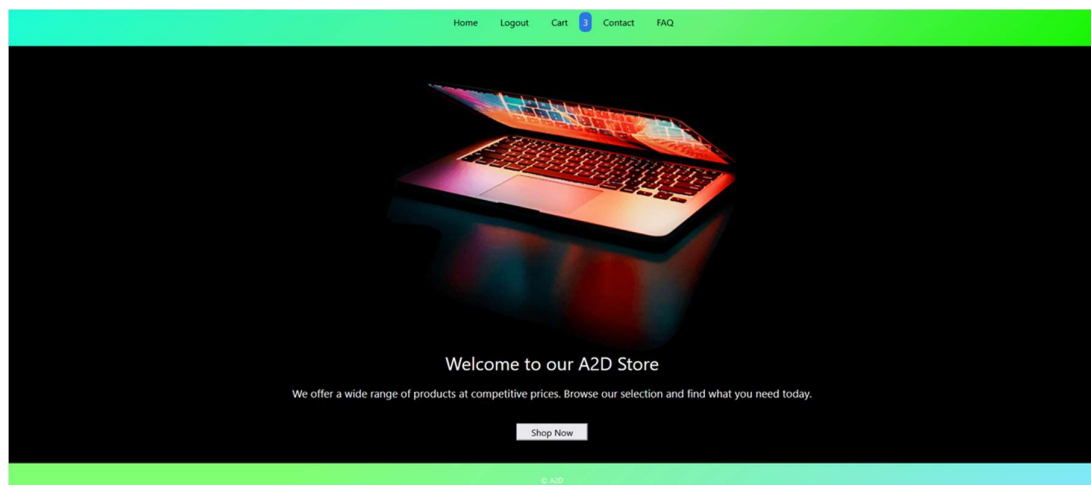


Figure 3.2: Home Page of Online Hardware Store

3.5.2 Login Page

Figure 3.3 provides the interface for login page of the website. A login page is of extreme importance to web design, especially for online stores or e-commerce websites. A creative and attractive login page will quickly catch the user's attention, direct a high volume of visitors to your website, and improve the user experience. Most login pages include elements such as username, password, and a highlighted CTA. A creative and attractive login page will quickly catch the user's attention, direct a high volume of visitors to your website, and improve the user experience. Most login pages include elements such as username, password, and a highlighted CTA.

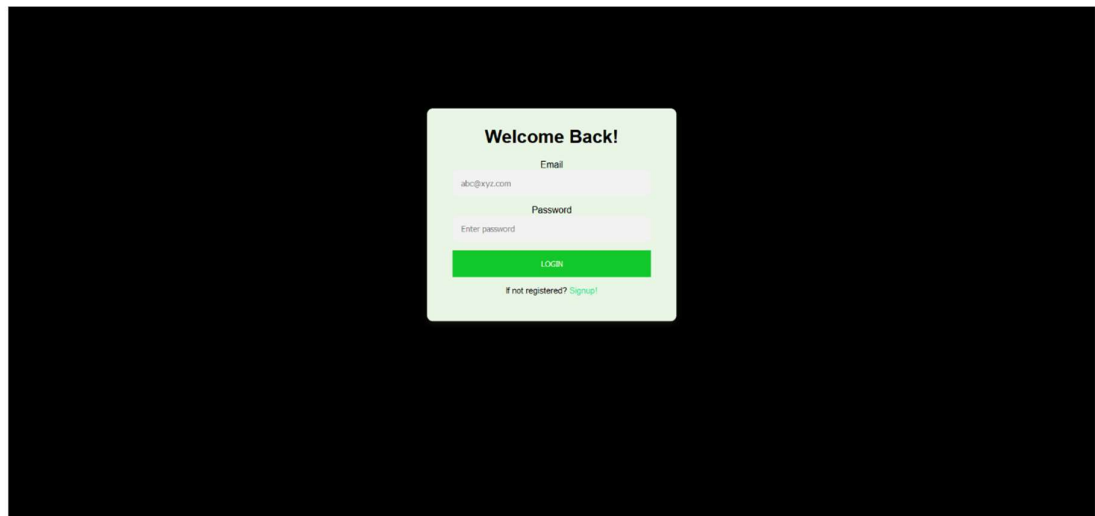


Figure 3.3: Login Page of Online Hardware Store

3.5.3 Signup Page

Figure 3.4 provides the interface for new user registration page of the website. A signup page is a page on your website where users can sign up to use your product. They're designed to capture the email addresses of visitors and make it easy for them to access your product. Usually, a signup page contains a form (a signup form), where users can register by providing necessary information, such as their Username and Password.

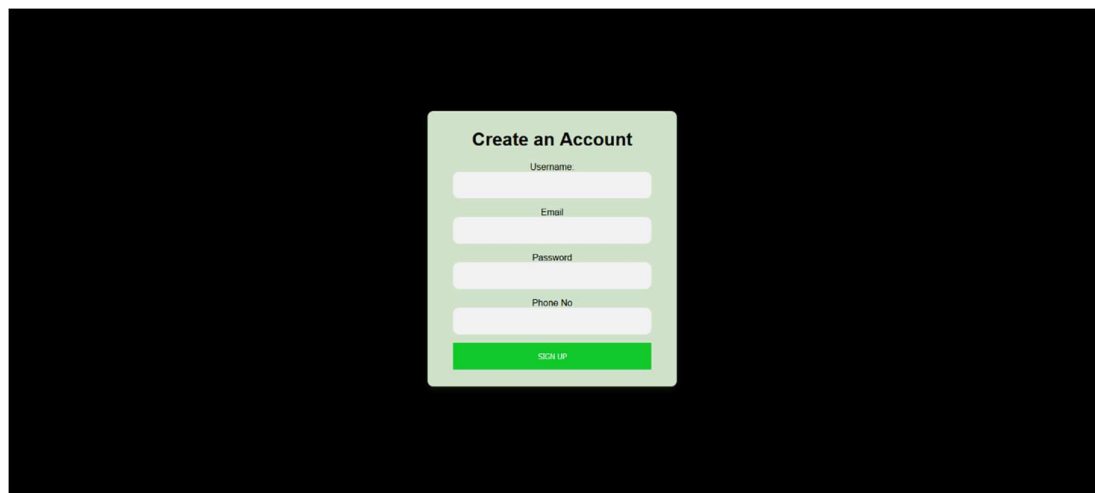


Figure 3.4: Signup Page of Online Hardware Store

3.5.4 Product Details Page

Figure 3.5 provides the detailed information about a hardware. A product detail page is a web page on an online hardware store website that provides information on a specific product. This information includes size, color, price, shipping information, reviews, and other relevant information customers want to know before purchasing. Product details are displayed on a web page called a product detail page, which helps customers make purchasing decisions.

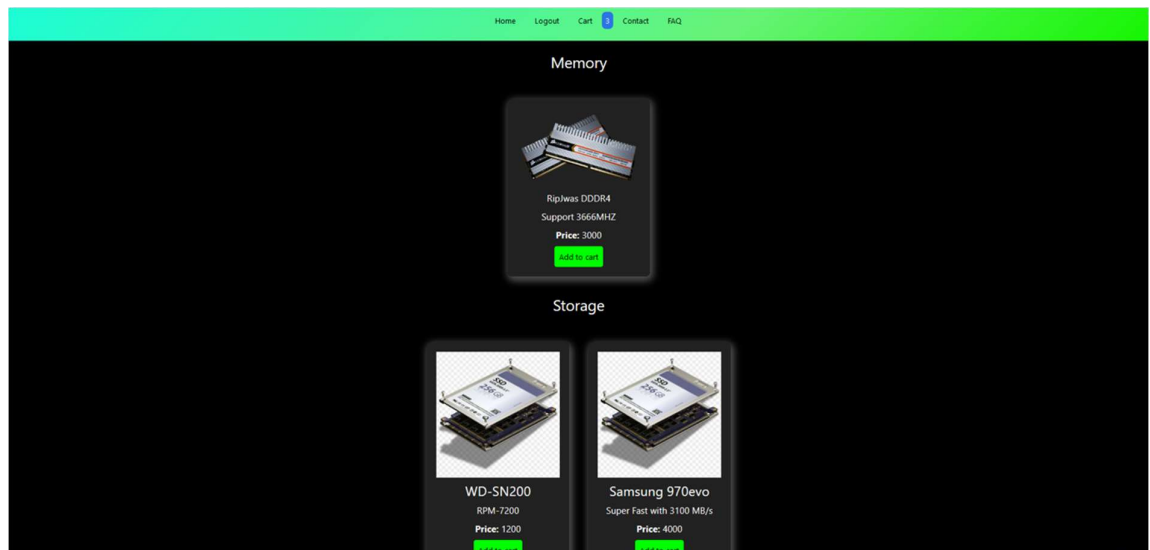


Figure 3.5: Product Details Page

3.5.5 Cart Page

Figure 3.6 provides the cart page of the website. A cart page is an essential part of an e-commerce website. It is the page where users can pile up what they want to buy from the website and then simply checkout by paying online.




Home Logout Cart 3 Contact FAQ					
S.NO	Product Name	Product Image	Description	Price	Action
1	RipJwas DDR4		Support 3666MHZ	3000	Delete
2	Samsung 970evo		Super Fast with 3100 MB/s	4000	Delete
3	ROG Mother Board		Suppoerts all ports	43000	Delete
				Empty cart Shop More Check out	Grand Total:Rs 50000

Figure 3.6: Cart Page

3.5.5 FAQ Page

Figure 3.7 provides the FAQ summary page of the website. An FAQ page or section is a specific page or a section within your Online hardware store where you list down all the questions your customers might have or ask them frequently regarding your business, products/services, policies, processes, and more. It can reduce the anxiety of customers about buying things online and builds trust between you and your customers.

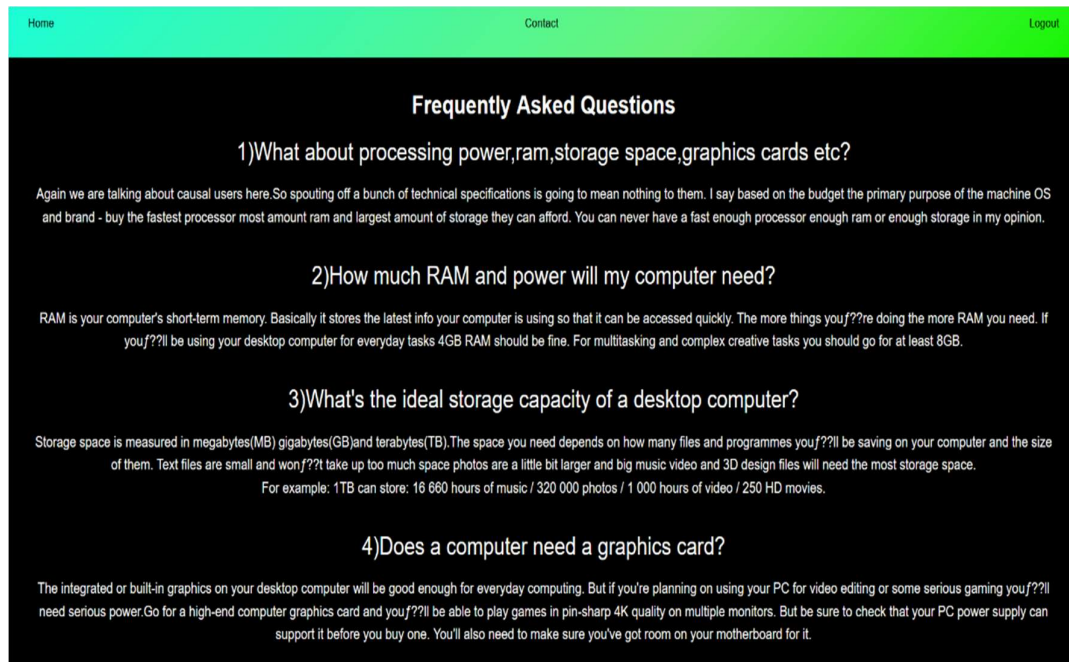


Figure 3.7: FAQ Page

3.5.6 Admin Page

Figure 3.8 provides the admin page of the website.An admin page is a page on a website that allows authorized users to manage the website's content and settings. It is typically only accessible to users with administrative privileges and is used to perform tasks such as managing user accounts, setting site-wide preferences, and publishing content.

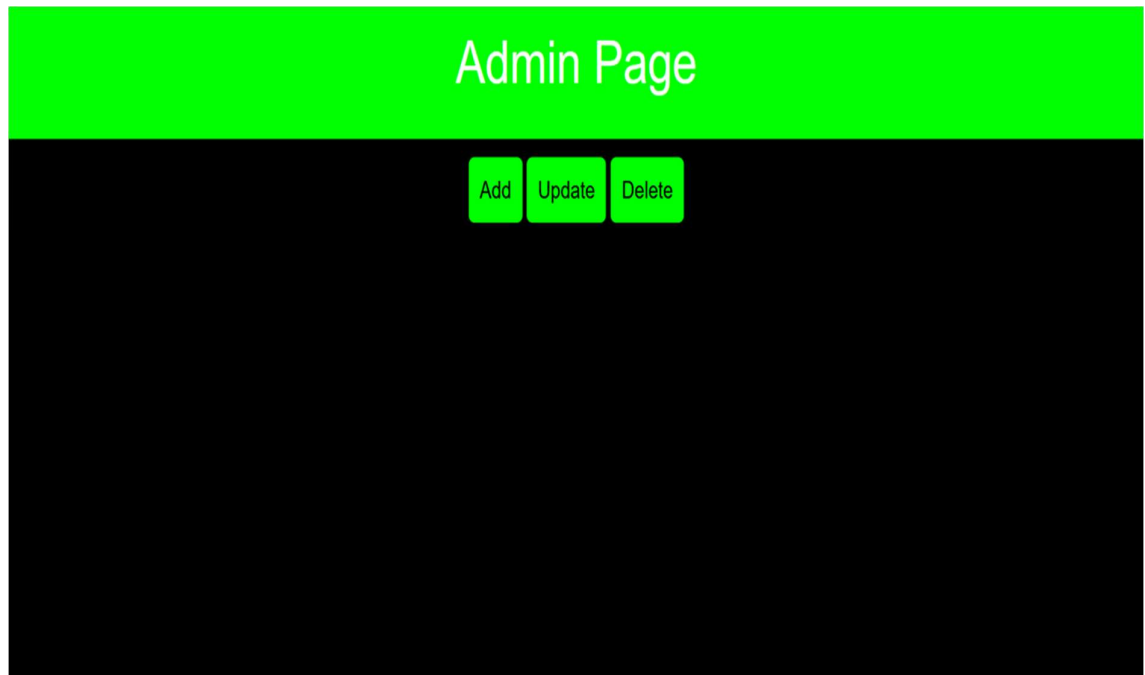
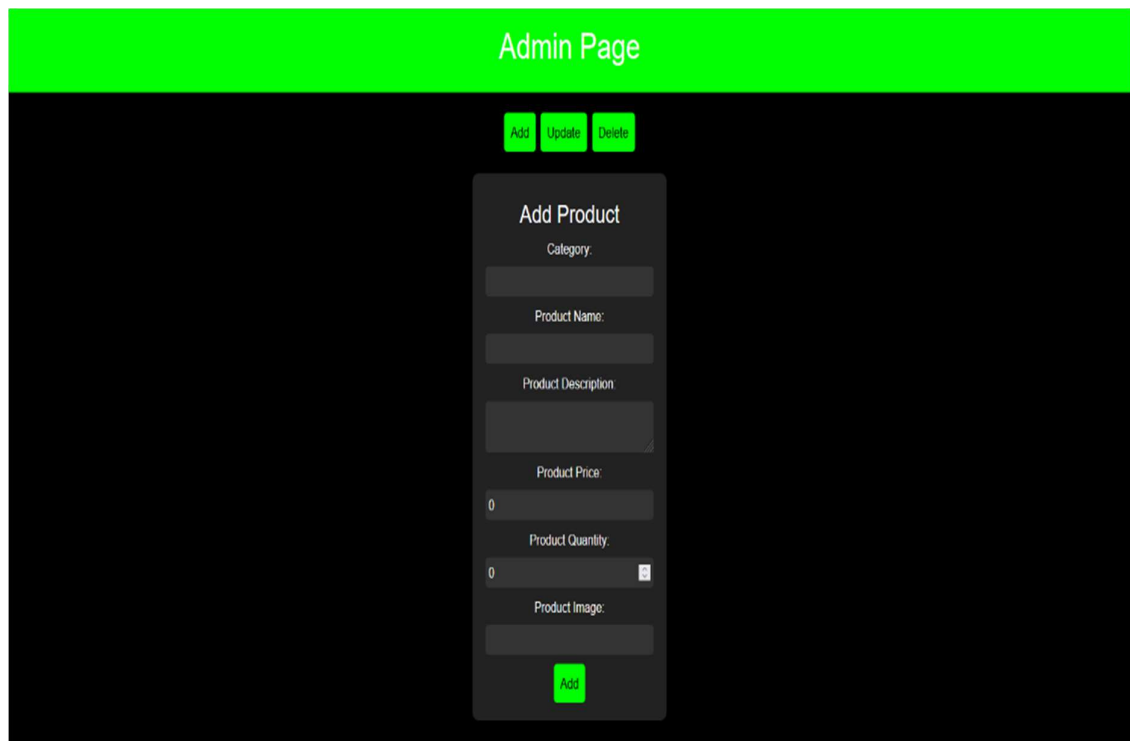


Figure 3.8: Admin Page

3.5.7 Add Product Page

Figure 3.9 provides the add product page of the website. An Add Product Page on an online store website is a page where the website owner or administrator can add new products to their online store. This page typically includes fields for entering product information such as name, description, price, images, and other relevant details.

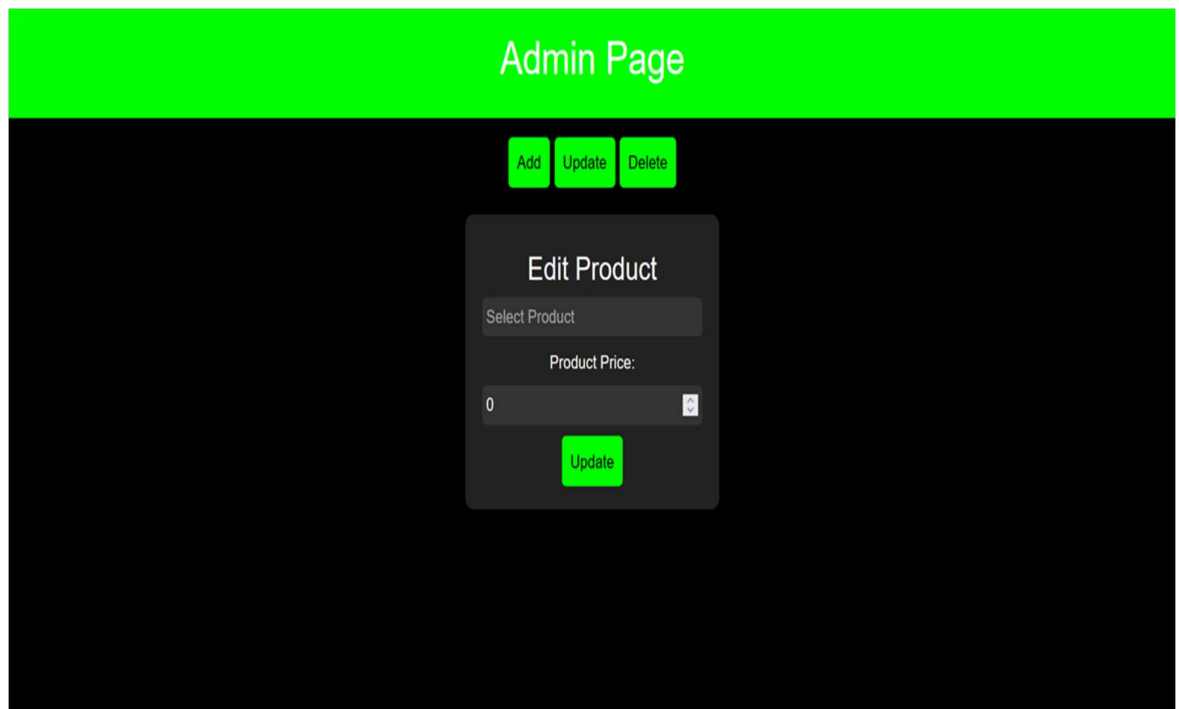


The image shows a web interface for an admin page. At the top, there is a green header bar with the text "Admin Page" in white. Below the header, there is a dark gray background. In the center, there is a white rectangular form titled "Add Product". Above the form, there are three small green buttons labeled "Add", "Update", and "Delete". The form contains several input fields: "Category:" (a text input), "Product Name:" (a text input), "Product Description:" (a text area), "Product Price:" (a text input with the value "0"), "Product Quantity:" (a text input with the value "0" and a small square icon to its right), and "Product Image:" (a text input). At the bottom of the form, there is a small green button labeled "Add".

Figure 3.9: Add product Page.

3.5.8 Edit Product Page

Figure 3.10 provides the edit page of the website. An Edit Product Page on an online store website is a page where the website owner or administrator can edit details products to their online store. This page typically includes fields for entering product information such as name, description, price, images, and other relevant details.



The image shows a web interface for an admin page. At the top, there is a green header bar with the text "Admin Page" in white. Below the header, there are three green buttons labeled "Add", "Update", and "Delete". In the center, there is a dark gray modal box titled "Edit Product". Inside this modal, there is a text input field labeled "Select Product", a label "Product Price:" followed by a text input field containing the number "0" and a small downward arrow icon, and a green "Update" button at the bottom.

Figure 3.10: Edit Page

3.5.9 Delete Page

Figure 3.11 provides the delete page of the website. A Delete Product Page on an online store website is a page where the website owner or administrator can delete products details of their online store. This page typically includes fields for entering product information such as name, description, price, images, and other relevant details.

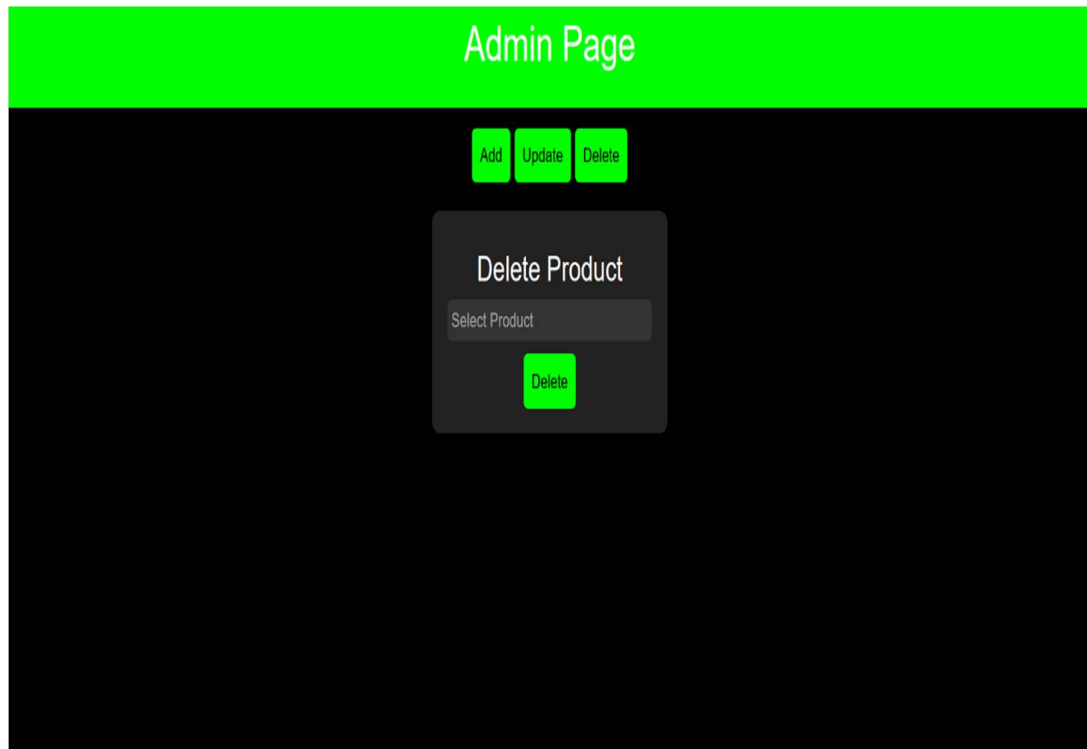


Figure 3.11: Delete Page

Conclusion:

Based on the user interface screenshots presented, it can be concluded that the design of the online hardware store's interface is visually appealing and user-friendly. The use of high-quality product images, clear typography, and a consistent color scheme creates a cohesive and professional look. The layout of the homepage effectively showcases the store's featured products, while the top navigation bar and search bar make it easy for users to find what they are looking for. The product category pages are well-organized and provide useful filtering options, allowing users to quickly narrow down their search. The product pages themselves are well-designed, with detailed product descriptions and clear pricing information. The "Add to Cart" and "Check Out" buttons are prominently displayed, making it easy for users to complete their purchase. The shopping cart and checkout pages are also well-designed, with clear instructions and easy-to-fill forms. The use of a progress bar during checkout provides users with a clear understanding of the steps required to complete their purchase. Overall, the user interface screenshots demonstrate that the online hardware store's interface is well-designed and user-friendly, creating a positive user experience.

CHAPTER 4

SYSTEM IMPLEMENTATION

4.1 LOGIN IMPLEMENTATION

The login credentials are obtained. If the credentials are OK, then the user is redirected to the homepage

```

POST email id, password
    IF userid, password valid
        RETURN homepage
    ELSE
        TOAST Invalid Credential
  
```

4.2 SIGN-UP IMPLEMENTATION

The form fields are obtained. If they are valid, then the user is added to the database.

```

POST requestFields
    IF requestFields valid
        RETURN added to db
    ELSE
        TOAST enter valid details
  
```

4.3 ORDER IMPLEMENTATION

The form fields are obtained. If they are valid, then the computer product is added to the database.

```

GET Product
    IF Productavailable valid
        RETURN Order page
    ELSE
        RETURN Out of Stock
  
```

4.4 ADD TO CART IMPLEMENTATION

The form fields are obtained. If computer product stock is available, then the POST Product in cart

IF Product in Cart valid

RETURN Confirm Order

ELSE

RETURN Item not found

4.5 PRODUCT IMPLEMENTATION

The form fields are obtained. If computer product stock is available, then the GET Product

IF USER is valid

RETURN Products

ELSE

RETURN Null

4.6 ADD PRODUCT IMPLEMENTATION

The form fields are obtained. If Admin Login is valid, then the POST Product

IF ADMIN is valid

RETURN Addproduct page

ELSE

RETURN Null

4.7 EDIT PRODUCT IMPLEMENTATION

The form fields are obtained. If Admin Login is valid, then the POST Product

IF ADMIN is valid

RETURN Editproduct page

ELSE

RETURN Null

4.8 DELETE PRODUCT IMPLEMENTATION

The form fields are obtained. If Admin Login is valid, then the POST Product

IF ADMIN is valid

 RETURN Deleteproduct page

ELSE

 RETURN Null

CHAPTER 5

RESULTS AND DISCUSSION

5.1 TEST CASES AND RESULTS

5.1.1 Test Cases and Results for Login function:

The Table 5.1, Table 5.2 shows that the possible test data for the both positive and negative test case given below, if the user is already having account then the output is true otherwise false.

Table 5.1: Positive Test Case and result for Login

If the user enters the credentials for login in correct way, that is when the user enters both the username and the password correctly at the same time, or else the user may not be able to login.

Test Case ID	TC1
Test Case Description	It tests whether the given login details are valid or not
Test Data	test@gmail.com, test@123
Expected Output	TRUE
Result	PASS

Table 5.2: Negative Test Case and result for Login

If the user enters the credentials for login in correct way, that is when the user enters both the username and the password correctly at the same time, or else the user may not be able to login.

Test Case ID	TC2
Test Case Description	It tests whether the given login details are valid or not
Test Data	test@gmail.com
Expected Output	FALSE
Result	PASS

5.1.2 Test Case and Results for Signup Function:

The Table 5.3, Table 5.4 shows that the possible test data for both positive and negative test case given below, if the user is already having account then the output is false otherwise true.

Table 5.3: Positive Test Case and result for Signup

When an user needs to create a new account he/she need to enter the asked details in a desired way as the system requesting them to the below test case is when the user satisfied all the test data and registered successfully.

Test Case ID	TC3
Test Case Description	It tests whether the given registration details are valid or not
Test Data	abishe, test@gmail.com, test@123, test@123
Expected Output	TRUE
Result	PASS

Table 5.4: Negative Test Case and result for Signup

When an user needs to create a new account he/she need to enter the asked details in a desired way as the system requesting them to the below test case is when the user who is not satisfied all the test data and the registration failed.

Test Case ID	TC4
Test Case Description	It tests whether the given registration details are valid or not
Test Data	abishek, test@gmail.com, test@123, test@123
Expected Output	FALSE
Result	PASS

5.1.3 Test Case and Results for Add to Cart:

The Table 5.5, Table 5.6 shows that the possible test data for both positive and negative test case given below, if the value of the add to cart fields are valid then the output is true otherwise false.

Table 5.5: Positive Test Case and result for cart

When the user needs to buy product from our website he need to select the product which he needs by placing them in the cart and then by that the user can confirm their order by checking out, However It is only possible is the stock is available

Test Case ID	TC7
Test Case Description	It tests whether the product can be added to cart or not
Test Data	Sample-hardware product, 10
Expected Output	TRUE
Result	PASS

Table 5.6: Negative Test Case and result for Add to cart.

Test Case ID	TC8
Test Case Description	It tests whether the product can be added to cart or not

Test Data	Sample-hardware product, 10000
Expected Output	FALSE
Result	PASS

5.1.2 Test Case and Results for Admin signup Function:

The Table 5.3, Table 5.4 shows that the possible test data for both positive and negative test case given below, if the user is already having account then the output is false otherwise true.

Table 5.3: Positive Test Case and result for Admin login

Test Case ID	TC3
Test Case Description	It tests whether the given login details are valid or not
Test Data	Admin,admin@gmail.com, admin@123
Expected Output	TRUE
Result	PASS

Table 5.4: Negative Test Case and result for Admin login

Test Case ID	TC4
Test Case Description	It tests whether the given login details are valid or not
Test Data	Admin,admin@gmail.com, admin@
Expected Output	FALSE
Result	PASS

5.2 Results

5.1.1 Login Page

Figure 3.12 provides the interface for login page of the website. A login page is of extreme importance to web design, especially for online stores or e-commerce websites. A creative and attractive login page will quickly catch the user's attention, direct a high volume of visitors to your website, and improve the user experience. Most login pages include elements such as username, password, and a highlighted CTA. A creative and attractive login page will quickly catch the user's attention, direct a high volume of visitors to your website, and improve the user experience. Most login pages include elements such as username, password, and a highlighted CTA.

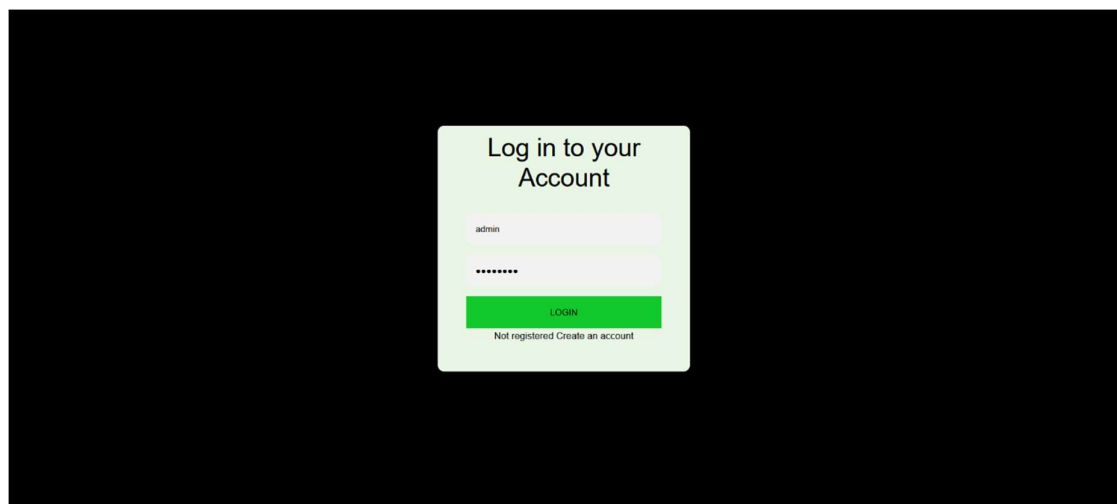


Figure 3.12: Login Page of Online Hardware Store

5.1.2 Signup Page

Figure 3.13 provides the interface for new user registration page of the website. A signup page is a page on your website where users can sign up to use your product. They're designed to capture the email addresses of visitors and make it easy for them to access your product. Usually, a signup page contains a form (a signup form), where users can register by providing necessary information, such as their Username and Password.

Figure 3.13: Signup Page of Online Hardware Store

5.1.2 Add Product Page

Figure 3.14 provides the add product page of the website. An Add Product Page on an online store website is a page where the website owner or administrator can add new products to their online store. This page typically includes fields for entering product information such as name, description, price, images, and other relevant details.

Figure 3.14: Add product Page.

5.1.3 Edit Product Page

Figure 3.15 provides the edit page of the website. An Edit Product Page on an online store website is a page where the website owner or administrator can edit details products to their online store. This page typically includes fields for entering product information such as name, description, price, images, and other relevant details.

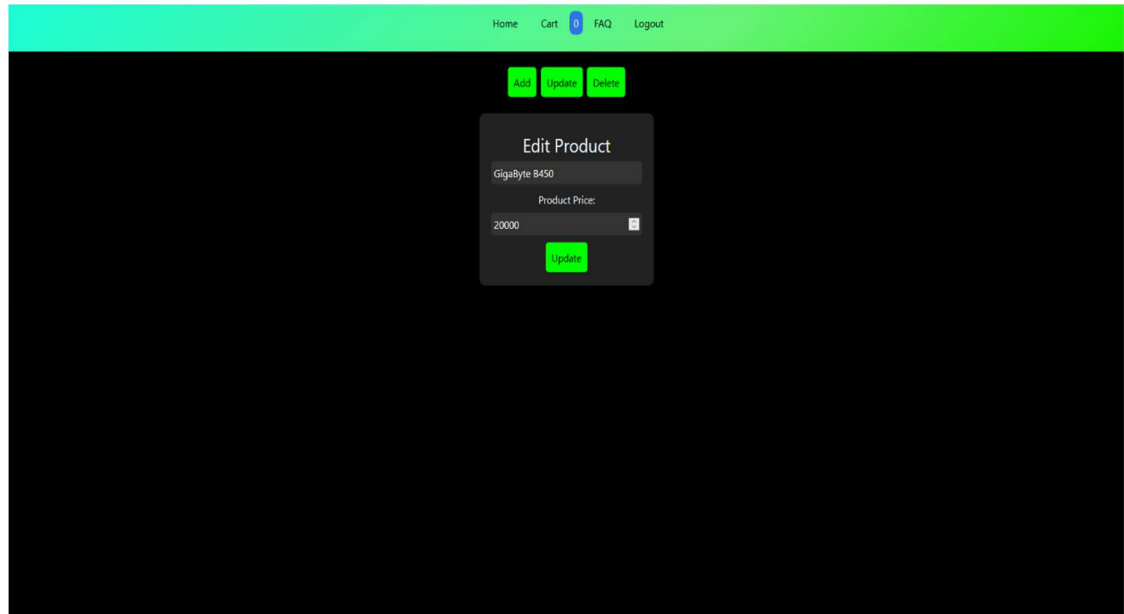


Figure 3.15: Edit Page

5.1.4 Delete Page

Figure 3.16 provides the delete page of the website. A Delete Product Page on an online store website is a page where the website owner or administrator can delete products details of their online store. This page typically includes fields for entering product information such as name, description, price, images, and other relevant details.

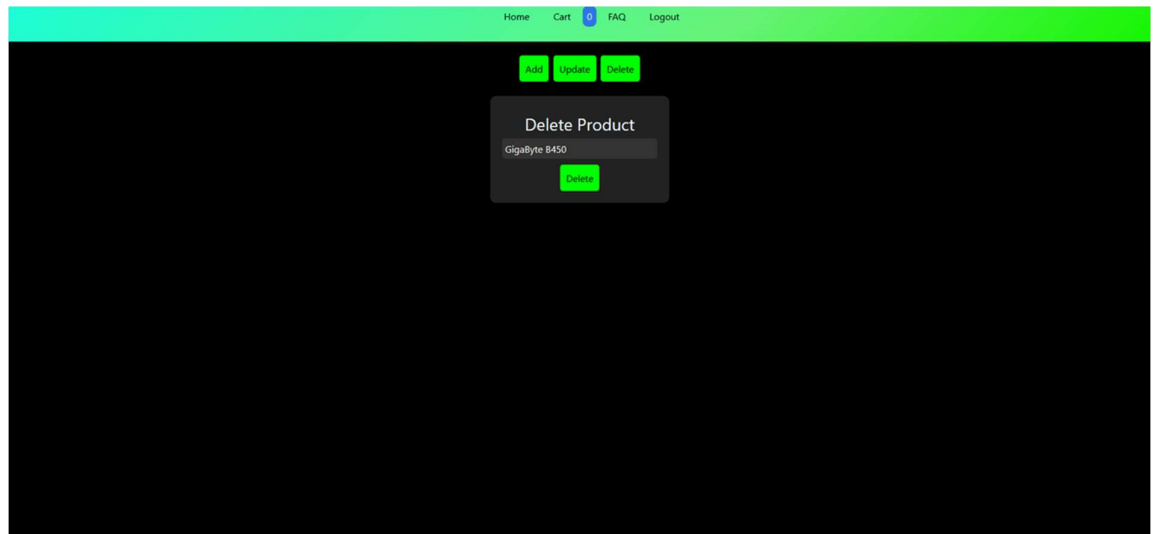


Figure 3.16: Delete Page

CHAPTER 6

CONCLUSION AND FUTURE ENHANCEMENT(S)

In conclusion, the Online hardware Store project has been successfully implemented and offers a user-friendly and efficient platform for customers to purchase hardware online. The website offers a wide range of products, from RAM to CPU, and provides customers with detailed product information, including images, specifications, and customer reviews. The payment and checkout process are simple and secure, with multiple payment options available to customers. The project has been developed using modern web technologies, ensuring that the website is responsive, fast, and accessible across different devices.

APPENDIX – A

SYSTEM REQUIREMENTS

HARDWARE REQUIREMENT:

Server	:	Any server with 10 TB Storage and 128 GB of Memory
Processor	:	Intel Core i3 10 th Gen or above
Memory	:	Minimum 8 GB
Storage	:	Minimum 256 GB of Disk Storage
Network	:	Minimum 10 MBPS Bandwidth

SOFTWARE REQUIREMENT:

Operating System	:	Any
DBMS	:	MongoDB
IDE used	:	Visual Studio Code 2023
Angular CLI Version	:	15.2.4

APPENDIX – B

SOURCE CODE

login.component.html

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

    <form #loginForm="ngForm" (ngSubmit)="login(loginForm.value)">

        <h1>Log in to your Account</h1>

        <br>

        <input type="text" id="utextbox" placeholder="Username" name="name"
        [(ngModel)]="user.name"/>

        <br>

        <input type="password" id="passwd-text-box" placeholder="Password"
        name="pass" [(ngModel)]="user.pass"/>

        <br/>

        <button class="button" onclick="login()"><a
        routerLink="/home">Login</a></button>

    </form>

    <p>

        Not registered

        <a class="sign" routerLink="/signup">Create an account</a>

    </p>

</div>
```

login.component.ts

```
import { HttpClient } from '@angular/common/http';
import { Component, OnInit } from '@angular/core';
import { LoginService } from '../login.service';
import { Router } from '@angular/router';
```

```

export class User{
  constructor(public name:String,public pass:String){}
}

@Component({
  selector: 'app-login',
  templateUrl: './login.component.html',
  styleUrls: ['./login.component.css']
})
export class LoginComponent implements OnInit {
  public submitted!:Boolean;
  public user:User = new User(",");
  public res:any;
  constructor(private http:HttpClient,private router:Router){}
  ngOnInit(): void { }
  public login(data:User)
  {
    this.submitted=true;
    this.user=data;
    console.log(this.user);
    const url = "http://localhost:5555/api/user_login";
    this.http.post(url, data).subscribe((data)=>{
      console.log(data);
      this.res = data;
      if(this.res.auth == "true"){
        LoginService.setId(this.res.name,this.res._id);
        this.router.navigateByUrl('/home');
      }
    })
  }
}

```

```

    });
  }
}

```

signup.component.html

```

<div class="login-container" style="width: 500px;">
  <form #signupForm="ngForm" (ngSubmit)="signup(signupForm.value)">
    <h1>Sign up</h1>
    <br>
    <input type="text" id="utextbox" name="name" placeholder="Username"
      [(ngModel)]="user.name" />
    <input type="password" id="passwd-text-box" name="pass"
      placeholder="Password" [(ngModel)]="user.pass"/>
    <input type="password" id="passwd-text-box" name="cpass"
      placeholder="Confirm Password" [(ngModel)]="cpass"/>
    <button class="button" onclick="signup()"><a
      routerLink="/home">Signup</a></button>
  </form>
</div>

```

signup.component.ts

```

import { Component } from '@angular/core';
import { HttpClient } from '@angular/common/http';
import { Router } from '@angular/router';
export class User {
  constructor(public name:String,public pass:String){}
}
@Component({
  selector: 'app-signup',
  templateUrl: './signup.component.html',
  styleUrls: ['./signup.component.css']
})
export class SignupComponent {

```

```

public res:any;

public cpass:any;

public user= new User(",");

constructor(private http:HttpClient,private router:Router){}

public signup(data:User){

  if(this.user.pass==this.cpass){

    console.log("Corrct Password");

    console.log(this.user);

    const url = "http://localhost:5555/api/sign_up";

    this.http.post(url, this.user).subscribe((data)=>{

      console.log(data);

      this.res = data;

    });

    this.router.navigateByUrl('/login');

  }

  else{

    console.log("Check Password");

  }

}

}

```

header.component.html

```

<header>

  <nav>

    <ul>

      <div class="logo">

        </div>

        <a routerLink="/home">Home</a>

        <a routerLink="/">Logout</a>

        <a routerLink="/cart">Cart</a>

```

```

    <span>{{totalitem}}</span>
    <a routerLink="/faq">FAQ</a>
  </ul>
</nav>
</header>

```

header.component.ts

```

import { Component,OnInit } from '@angular/core';
import { CartService } from '../cart.service';

@Component({
  selector: 'app-header',
  templateUrl: './header.component.html',
  styleUrls: ['./header.component.css']
})
export class HeaderComponent implements OnInit{
  public totalitem :number =0;
  constructor(private cartservice:CartService){}
  ngOnInit(): void {
    this.cartservice.getProducts()
    .subscribe(res=>{
      this.totalitem =res.length;
    })
  }
}

```

home.component.html

```

<app-header></app-header>

<main>

  <h1>Welcome to our A2D Store</h1>

```


<p>We offer a wide range of products at competitive prices. Browse our selection and find what you need today.</p>

<button onclick="list()">Shop Now</button>

</main>

<footer>

<p>© A2D</p>

</footer>

home.component.ts

```
import { Component,OnInit } from '@angular/core';
import { HttpClient } from '@angular/common/http';
import { CartService } from '../cart.service';
// import { ApiService } from 'src/app/service/api.service';
export interface Seed{
  name:String;
  _id:number;
  category:String;
  description:String;
  unit:String;
  quantity:number;
}
@Component({
  selector: 'app-home',
  templateUrl: './home.component.html',
  styleUrls: ['./home.component.css']
})
```

```
export class HomeComponent implements OnInit {
  constructor(private cartservice :CartService){}
  ngOnInit(): void {
```

```

    this.cartservice.getProducts()

    .subscribe()

  }

  addtoCart(item:any){

  }

}

```

product.component.html

```

<app-header></app-header>

  <br>

  <div class="container1" >

    <div class="category">

      <h1>Memory</h1>

      <div class="category1">


        <br>

        <ng-container class="cards" *ngFor="let item of productList " >

          <div class="product" *ngIf="item.category == 'RAM'" >

            <img src={{item.imgurl}} alt="Product 1">

            <p>{{item.name}}</p>

            <p>{{item.description}}</p>

            <p><strong>Price:</strong> {{item.price}}</p>

            <input type="hidden" value="{{item._id}}"/>

            <button class="btn btn-primary" (click)="addtocart(item)">Add to
cart</button>

          </div>

        </ng-container></div>

        <br>

        <h1>Storage</h1>

        <div class="category1">

```

```

<ng-container *ngFor="let item of productList ">
  <div class="product" *ngIf="item.category == 'hdd'" >
    <img src={{item.imgurl}} alt="Product 2">
    <h3>{{item.name}}</h3>
    <p>{{item.description}}</p>
    <p><strong>Price:</strong> {{item.price}}</p>
    <input type="hidden" value="{{item._id}}"/>
    <button class="btn btn-primary" (click)="addtocart(item)">Add to
cart</button>
  </div>
</ng-container></div>

<h1>Processor</h1>
<div class="category1">
  <ng-container *ngFor="let item of productList ">
    <div class="product" *ngIf="item.category == 'Processor'" >
      <img src={{item.imgurl}} alt="Product 3">
      <h3>{{item.name}}</h3>
      <p>{{item.description}}</p>
      <p><strong>Price:</strong> {{item.price}}</p>
      <input type="hidden" value="{{item._id}}"/>
      <button class="btn btn-primary" (click)="addtocart(item)">Add to
cart</button>
    </div>
  </ng-container></div>

  <h1>Mother Board</h1>
  <div class="category1">
    <ng-container *ngFor="let item of productList ">
      <div class="product" *ngIf="item.category == 'board'" >
        <img src={{item.imgurl}} alt="Product 4">
        <h3>{{item.name}}</h3>

```

```

        <p>{{item.description}}</p>
        <p><strong>Price:</strong> {{item.price}}</p>
        <input type="hidden" value="{{item._id}}" />
        <button class="btn btn-primary" (click)="addtocart(item)">Add to
cart</button>
    </div>
</ng-container></div>
</div>
</div>

```

product.component.ts

```

import { Component, OnInit } from '@angular/core';
import { HttpClient } from '@angular/common/http';
import { CartService } from '../cart.service';
@Component({
  selector: 'app-product',
  templateUrl: './product.component.html',
  styleUrls: ['./product.component.css']
})
export class ProductComponent {

  public productList:any;
  empty = false;
  public item: any;
  constructor(private http: HttpClient ,private cartservice :CartService ){ }
  ngOnInit(): void {
    this.http.get('http://localhost:5555/api/products').subscribe(
      Response=>{
        if(Response){
          console.log("data received");
          this.productList=Response;

```

```

    }
    console.log(this.productList);
  }
)
}
addtocart(item:any){
  this.cartservice.addtocart(item);
}
}

```

faq.component.html

fag html

```

<!DOCTYPE html>

<html>

<head>

  <title>Frequently Asked Questions</title>

</head>

<body>
  <header>
    <nav>
      <ul>
        <li><a routerLink="/home">Home</a></li>

        <li><a routerLink="/contact">Contact</a></li>

        <li><a routerLink="/">Logout</a></li>

      </ul>
    </nav>

```

</header>

<main>

<h1> Frequently Asked Questions</h1>

<h1>1)What about processing power,ram,storage space,graphics cards etc?
</h1>

<p>Again we are talking about casual users here. So spouting off a bunch of technical specifications is going to mean nothing to them. I say based on the budget the primary purpose of the machine OS and brand - buy the fastest processor most amount ram and largest amount of storage they can afford. You can never have a fast enough processor enough ram or enough storage in my opinion. </p>

<h1>2)How much RAM and power will my computer need?</h1>

<p>RAM is your computer's short-term memory. Basically it stores the latest info your computer is using so that it can be accessed quickly. The more things you're doing the more RAM you need. If you'll be using your desktop computer for everyday tasks 4GB RAM should be fine. For multitasking and complex creative tasks you should go for at least 8GB.</p>

<h1>3)What's the ideal storage capacity of a desktop computer?</h1>

<p>Storage space is measured in megabytes(MB) gigabytes(GB)and terabytes(TB).The space you need depends on how many files and programmes you'll be saving on your computer and the size of them. Text files are small and won't take up too much space photos are a little bit larger and big music video and 3D design files will need the most storage space.

For example: 1TB can store: 16 000 hours of music / 320 000 photos / 1 000 hours of video / 250 HD movies. </p>

<h1>4)Does a computer need a graphics card?

</h1>

<p>The integrated or built-in graphics on your desktop computer will be good enough for everyday computing. But if you're planning on using your PC for video

editing or some serious gaming you'll need serious power. Go for a high-end computer graphics card and you'll be able to play games in pin-sharp 4K quality on multiple monitors. But be sure to check that your PC power supply can support it before you buy one. You'll also need to make sure you've got room on your motherboard for it.

<h1>5) Should I get an all-in-one desktop computer or a tower computer?</h1>

<p>Tower computers are still popular but they can be fairly bulky. If you're limited in terms of space think about going for an all-in-one or mini-PC. All-in-one desktop computer: The computer is built into the monitor so you don't have to find space for a tower and separate monitor. Mini-PC: These are so compact they can fit almost anywhere. This makes mini-PCs a great solution when space is really tight. Tower: This traditional desktop PC is made up of a tower (the case that holds all the components) and separate desktop computer monitor. They're popular with gamers that need a powerful rig and with families looking for an affordable PC.</p>

</main>

<footer>

<p>© A2D</p>

</footer>

</body>

</html>

faq.component.ts

```
import { Component } from '@angular/core';
```

```
@Component({
```

```
  selector: 'app-faq',
```

```
  templateUrl: './faq.component.html',
```

```
  styleUrls: ['./faq.component.css']
```

```
})
```

```
export class FaqComponent {
```

```
}
```

admin.component.html

```

<body>
  <div class="header">
    <h1>Admin Page</h1>
  </div>
  <ng-container>
    <div class="options">
      <button (click)="b1()">Add</button>
      <button (click)="b2()">Update</button>
      <button (click)="b3()">Delete</button>
    </div>
  </ng-container>
  <div class="center">
    <ng-container class="add" *ngIf="flag1">
      <div class="product">
        <h3>Add Product</h3>
        <form #addForm="ngForm"
          (ngSubmit)="addproduct(addForm.value)">
          <label for="category">Category:</label>
          <input type="text" id="name" name="name"
            [(ngModel)]="user.category">
          <label for="name">Product Name:</label>
          <input type="text" id="name" name="name"
            [(ngModel)]="user.name">
          <label for="description">Product Description:</label>
          <textarea id="description" name="description"
            [(ngModel)]="user.desc"></textarea>
          <label for="price">Product Price:</label>
          <input type="text" id="price" name="price"
            [(ngModel)]="user.price">
          <label for="price">Product Quantity:</label>
          <input type="number" id="price" name="price"
            [(ngModel)]="user.quantity">

```



```

        <label for="image">Product Image:</label>
        <input type="text" id="image" name="image"
[(ngModel)]="user.imgurl">
        <button (click)="addproduct">Add</button>
    </form>
</div>
</ng-container>
<ng-container class="add" *ngIf="flag2">
    <div class="product">
        <h3>Edit Product</h3>
        <form #updateForm="ngForm"
(ngSubmit)="update(updateForm.value)">
            <input type="text" name="name" placeholder="Select Product" list="name"
[(ngModel)]="updata.name">
            <datalist id="name">
                <ng-container *ngFor="let item of productList ">
                    <option value={{item.name}}>
                </ng-container>
            </datalist>
            <label for="price">Product Price:</label>
            <input type="number" id="price" name="price"
[(ngModel)]="updata.price">
            <button >Update</button>
        </form>
    </div>
</ng-container>
<ng-container class="add" *ngIf="flag3">
    <div class="product">
        <h3>Delete Product</h3>
        <form #deleteForm="ngForm"
(ngSubmit)="Delete(deleteForm.value)">

```

```

        <input type="text" name="fname" placeholder="Select Product" list="name"
[(ngModel)]="delData.fname">
        <datalist id="name">
            <ng-container *ngFor="let item of productList ">
                <option value={{ item.name }}>
            </ng-container>
        </datalist>

            <button>Delete</button>

        </form>

    </div>
</ng-container>
</div>
</body>

```

admin.component.ts

```

import { Component } from '@angular/core';
import { HttpClient } from '@angular/common/http';
import { Router } from '@angular/router';
import { UserService } from '../user.service';

export class User {
    constructor(public category:String,public name:String,public desc:String,public
price:number,public quantity:number,public imgurl:String){}
}

export class udata {
    constructor(public name:any,public price:number){}
}

```

```
export class DelData{
  constructor(public fname:String){}
}
```

```
@Component({
  selector: 'app-admin',
  templateUrl: './admin.component.html',
  styleUrls: ['./admin.component.css']
})
export class AdminComponent {
  public updata:udata = new udata("",0);
  public delData:DelData=new DelData("");
  public category: any;
  public name : any;
  public desc : any;
  public price!: number;
  public quantity !: number;
  public imgurl :any;
  public flag1 =0;
  public flag2 =0;
  public flag3 =0;
  public user= new User(",","0,0,");
  public res!: Object;
  constructor(private http:HttpClient,private router:Router){}
  productList:any;
  ngOnInit(): void {
    this.http.get('http://localhost:5555/api/products').subscribe(
```

```

    Response=>{
      if(Response){
        console.log("data received");

      }
      this.productList=Response;
      console.log(this.productList);
    }
  )
}

addproduct(data:User){
  {
    console.log(this.user);
    const url = "http://localhost:5555/api/addproduct";
    this.http.post(url, this.user).subscribe((data)=>{
      console.log(data);
      this.res = data;
    });
    this.router.navigateByUrl('/product');
  }
}

update(data:any){
  {
    console.log(this.updata);
    const url = "http://localhost:5555/api/productups";
    this.http.post(url, this.updata).subscribe((data)=>{
      console.log(data);
      this.res = data;
    });
  }
}

```

```

    });
    this.router.navigateByUrl('/product');
  }
}

```

```

Delete(data:any){
  {
    console.log(this.delData);
    const url = "http://localhost:5555/api/productdelete";
    this.http.post(url, this.delData).subscribe((data)=>{
      console.log(data);
      this.res = data;
    });
    this.router.navigateByUrl('/product');
  }
}

```

```

b1(){
  console.log("b1")
  this.flag1 =1;
  this.flag2 =0;
  this.flag3 =0;
}

```

```

b2(){
  console.log("b2")
  this.flag1 =0;
  this.flag2 =1;
  this.flag3 =0;
}

```

```

b3(){
  console.log("b3")
  this.flag1 =0;
  this.flag2 =0;
  this.flag3 =1;
}

```

```

}

```

cart.component.html

```

<app-header></app-header>
<ng-container *ngIf="product.lenght ==0">
  <div class="container">
    <div class="card">
      <h5 class="card-title">
        My Cart
      </h5>
    </div>
    <div class="center">
      <img src="" alt="">
      <h4>Your cart is empty</h4>
      <h6>Add item to it new</h6>
      <button class="btn btn-primary" routerLink="/product">Shop Now</button>
    </div>
  </div>
</ng-container>

<ng-container *ngIf="product.lenght !=0">

```

```

<div class="container">
  <div class="card-table">
    <div class="cart-product">
      <table class="table table-responsive">
        <thead>
          <tr>
            <th>S.NO</th>
            <th>Product Name</th>
            <th>Product Image</th>
            <th>Description</th>
            <th>Price</th>
            <th>Action</th>
          </tr>
        </thead>
        <tbody>
          <tr *ngFor="let item of product; let i = index">
            <td>{{i+1}}</td>
            <td>{{item.name}}</td>
            <td></td>
            <td>{{item.description}}</td>
            <td>{{item.price}}</td>
            <td>
              <button (click)="removeitem(item)" class="btn btn-danger"
>Delete</button>
            </td>
          </tr>
        </tbody>
      </table>
    </div>
  </div>
</div>

```

```

        <td><button (click)="emptycart()" class="btn btn-danger" >Empty
cart</button></td>

        <td><button class="btn btn-primary" routerLink="/product">Shop
More</button></td>

        <td><button class="btn btn-success" (click)="makeOrder()">Check
out</button></td>

        <td><strong>Grand Total:Rs {{grandtotal}} </strong></td>

    </tr>

</table>

</div>

</div>

</div>

</ng-container>

```

cart.component.ts

```

import { Component,OnInit} from '@angular/core';
import { CartService } from '../cart.service';
import { HttpClient } from '@angular/common/http';
import { Router } from '@angular/router';
import { LoginService } from '../login.service';

@Component({
  selector: 'app-cart',
  templateUrl: './cart.component.html',
  styleUrls: ['./cart.component.css']
})
export class CartComponent implements OnInit{
  public product :any=[];
  public name :any;
  public total:any;
  public now:Date = new Date();
  public order:any;
  public grandtotal :number=0;

```



```

    constructor(private cartservice :CartService,private http:HttpClient,private
router:Router,private loginservice :LoginService){}

    ngOnInit():void{

        this.cartservice.getProducts()

        .subscribe(res=>{

            this.product = res;

            this.grandtotal =this.cartservice.gettotalprice();

        })

    }

    removeitem(item :any){

        this.cartservice.removecartitem(item);

    }

    emptycart(){

        this.cartservice.removeallcart()

    }

    getTotal(){

        this.total=0;

        for(var i=0;i<this.product.length;i++)

        {

            this.total += this.product[i].price;

        }

    }

    getuser(){

        this.name=this.loginservice.currentUser;

    }

    makeOrder(){

        console.log("Order processed");

        this.getTotal();

        this.getuser();

        console.log(this.name)

```

```

console.log(this.product);
console.log(this.total);
this.order = {
  "odate": this.now.toLocaleDateString(),
  "grandtotal": this.total,
  "custid": this.name,
  "orderList": this.product
}
this.http.post('http://localhost:5555/api/order', this.order).subscribe((res) => {
  this.router.navigateByUrl('/product');
});
}
}

```

Server.js

```

const express = require('express');
var router = express.Router();
const Customer = require('../models/customer');
const Product = require('../models/product')
const Payment = require('../models/payment')
const Order = require('../models/order')
const mongoose = require('mongoose')
mongoose.connect('mongodb://127.0.0.1:27017/a2d');
var db = mongoose.connection;
app = express();
router.post('/user_login', async (req, res, next) => {
  console.log("Request from webmaster");
  res.set({
    'Access-Control-Allow-Origin': '*'

```

```

    });
    try{
        const pass = req.body.pass;
        const name= req.body.name;
        console.log(pass);
        console.log(name);
        const opass = await Customer.find({pass:pass});
        console.log(opass[0].pass);
        if(name==opass[0].name && opass.length!=0){
            res.json({auth:'true',name:req.body.name,_id:opass[0]._id});
        }
        else{
            res.json({auth:'false'});
        }
    }catch(err){
        console.log(err);
        res.sendStatus(500);
    }
});

router.post("/sign_up",async function(req,res){
    console.log("sjsjd");
    var name = req.body.name;
    var pass = req.body.pass;
    var data = {
        "name": name,
        "pass":pass,
    }
    console.log("new "+name);
    db.collection('customers').insertOne(data,function(err, collection){

```

```

        if (err) throw err;

        console.log("Customer inserted Successfully");

    });

})

router.post("/productdelete", async function(req, res) {
    res.set({
        'Access-Control-Allow-Origin': '*',
    });

    console.log("Delete requested");

    var fname = req.body.fname;
    var data = {
        "name": fname,
    }

    console.log(data);

    try {
        const result = await Product.deleteOne({ name: fname });
        res.json({ msg: 'success' });
    } catch (err) {
        console.log(err);
        res.sendStatus(500);
    }
})

router.get("/products", async function(req, res, next) {
    res.set({
        'Access-Control-Allow-Origin': '*'
    });

    try {
        const products = await Product.find();

```

```

        console.log(products)
        res.json(products);
    } catch(err){
        console.log(err);
        res.sendStatus(500);
    }
});

router.get("/getorders",async function(req,res,next){
    res.set({
        'Access-Control-Allow-Origin': '*'
    });
    try{
        const orders = await Order.find();
        res.json( orders);
    } catch(err){
        console.log(err);
        res.sendStatus(500);
    }
});

router.post("/addproduct",async function(req,res){
    console.log("requested product insertion");
    var name = req.body.name;
    var category = req.body.category;
    var description = req.body.desc;
    var price = req.body.price;
    var quantity = req.body.quantity;
    var imgurl =req.body.imgurl
    console.log(name+category+description+price+quantity+imgurl);

```

```

var data = {
    "name": name,
    "category":category,
    "description":description,
    "price":price,
    "quantity":quantity,
    "imgurl":imgurl
}
//console.log("new "+data);
db.collection('products').insertOne(data,function(err, collection){
    if (err) throw err;
    console.log("Product inserted Successfully");

});
})

router.post("/productups",async function(req,res){
    console.log("requested product update");
    res.set({
        'Access-Control-Allow-Origin':'*',
    });
    var name=req.body.name;
    var price=req.body.price;

    console.log(name);
    db.collection("products").updateOne({name:req.body.name},{ $set: {price:req.body.price}}, function(err, res) {
        if (err) throw err;
    });
    res.json({msg:"success"});
}

```

```

    })

    router.post("/order", async function(req, res){
      console.log("Accepted orders");
      const order = new Order({
        odate: req.body.odate,
        grandtotal: req.body.grandtotal,
        custid: req.body.custid,
        orderList: req.body.orderList
      });
      console.log(order);
      try {
        await order.save();
        res.send(order);
      } catch (ex) {
        console.log(ex.message);
      }
    })

    module.exports = router;

```

index.js

```

var express=require("express");
var bodyParser=require("body-parser");
const mongoose = require('mongoose');
const Product = require('./models/product');
const customer = require('./models/customer');
var cors = require('cors');

```

```

mongoose.connect('mongodb://127.0.0.1:27017/a2d');
var db=mongoose.connection;
db.on('error', console.log.bind(console, "connection error"));
db.once('connected', ()=>{
    console.log("connection succeeded");
});
const route= require('./router/route');
const product = require("./models/product");
var app=express();
app.use(cors());
app.use(bodyParser.json());
app.use('/api',route);
app.use(express.static('public'));
app.use(bodyParser.urlencoded({
    extended: true
}));

app.post("/sign_up",async function(req,res){
    console.log("received");
    var name = req.body.name;
    var pass = req.body.pass;
    var data = {
        "name": name,
        "pass":pass,
    }
    console.log("new "+name);
    db.collection('customers').insertOne(data,function(err, collection){
        if (err) throw err;
        console.log("Customer inserted Successfully");
    });
}

```



```

    });
  })

  app.post("/updateproduct",async function(req,res){
    console.log("requested product Updation");

  });

  app.get('/',function(req,res){
    res.set({
      'Access-control-Allow-Origin': '*'
    });
  }
);
app.listen(5555,()=>{
  console.log("server listening at port 5555");
});

```

REFERENCES

1. Angular Documentation - <https://angular.io/docs>
2. Angular Material Component Dev Kit - <https://material.angular.io/cdk/categories>
3. TypeScript by Infosys Springboard- <https://infyspringboard.onwingspan.com/typescript>
4. Angular by Infosys Springboard - <https://infyspringboard.onwingspan.com/angular>
5. Angular for Beginners by freeCodeCamp - <https://youtu.be/3qBXWUpoPHo>
6. Node.js and Express Tutorial by freeCodeCamp - <https://youtu.be/Oe421EPjeBE>
7. Complete MongoDB Tutorial - <https://youtube.com/playlist?list=PL4cUxeGkcC9h77dJ-QJlwGlZITd4ecZOA>