TABLE OF CONTENTS

TABLE OF CONTENT

INTRODUCTION

CHAPTER 1

INTRODUCTION

A shopping cart is a fundamental component of e-commerce websites that facilitates online shopping by enabling customers to select and store items they wish to purchase. Comparable to a physical shopping cart in a brick-and-mortar store, the online shopping cart serves as a virtual container for products chosen by users during their browsing experience.

1.1 OBJECTIVE:

- The primary objective of a shopping cart is to provide a frictionless and user-friendly process for customers to add desired items and seamlessly proceed to checkout.
- It should enable users to easily review, edit, and finalize their product selections, ensuring a straightforward path from product discovery to purchase confirmation.
- The shopping cart aims to contribute to higher conversion rates by creating a user experience that minimizes abandoned carts.
- Through clear product information, transparent pricing, and efficient navigation, the shopping cart should encourage users to complete their transactions, reducing the likelihood of users abandoning their shopping journey.

SYSTEM SPECIFICATION

CHAPTER 2

SYSTEM SPECIFICATION

2.1 HARDWARE SPECIFICATION:

Processor Type	: Intel(R) Celeron(R) N4000 CPU
Speed	: 1.10 GHz
RAM	: 4.00 GB
Hard Disk	: 512 GB SSD
Monitor	: LCD

2.2 SOFTWARE SPECIFICATION:

Operating System	Windows 10 Home
Front End	React Js
Back End	Node Js

2.2.1 FRONT END:

React.js, or simply React, is an open-source JavaScript library designed for building user interfaces, particularly for single-page applications where the content is dynamically updated without the need for a full page reload. Developed and maintained by Facebook, React has gained widespread popularity in the web development community for its simplicity, efficiency, and component-based architecture. At the core of React's philosophy is the concept of reusable components. Developers break down the user interface into small, modular components, each responsible for a specific part of the application's UI. This component-based approach not only enhances code maintainability but also allows for the efficient development of complex applications by managing each piece independently.

One of React's standout features is its virtual DOM (Document Object Model). Instead of directly manipulating the actual DOM for every change, React creates a virtual representation of the DOM in memory

2.2.2 BACK END

Node.js is a powerful and versatile open-source, server-side JavaScript runtime environment that has revolutionized the way developers approach building scalable and highperformance web applications. Initially released in 2009 by Ryan Dahl, Node.js has since gained immense popularity due to its non-blocking, event-driven architecture and its ability to enable server-side JavaScript execution.

At its core, Node.js utilizes the V8 JavaScript engine, developed by Google, to execute code efficiently. One of Node.js's defining features is its asynchronous, event-driven nature. This means that it can handle a large number of simultaneous connections without getting bogged down by traditional blocking I/O operations. This makes Node.js particularly well-suited for applications that require real-time features, such as chat applications, online gaming platforms, and collaborative tools.

Node.js allows developers to use JavaScript for both client-side and server-side scripting, creating a unified development environment. This "JavaScript everywhere" approach streamlines the development process and promotes code reuse between the front-end and back-end components of an application. This unification not only enhances developer productivity but also contributes to a more seamless and consistent application architecture.

SYSTEM STUDY

CHAPTER 3

SYSTEM STUDY

3.1 EXISTING SYSTEM:

The existing content of the shopping cart project encompasses various aspects crucial for its functionality and usability. It includes features like user authentication, product CatLog browsing, cart management, checkout processes, payment integration, order tracking, and user profile management. The user interface design is evaluated, focusing on its responsiveness across different devices and its overall user experience. The backend architecture, comprised of database design, API endpoints, and server-side logic, is scrutinized for its efficiency and scalability. Technologies such as frontend frameworks/libraries (e.g., React, Angular) and backend frameworks (e.g., Express.js, Django) are discussed along with testing methodologies (unit, integration, and end-to-end testing) and performance considerations (optimization, caching, scalability). Security measures, compliance with web standards, and collaboration tools are also addressed to ensure the project's stability, security, and maintainability. The analysis culminates in recommendations for future enhancements and areas for further development, emphasizing the importance of continuous improvement in e-commerce platforms.

3.3.1 DRAWBACKS:

1.Complexity in managing various functionalities.

2.Scalability challenges with increasing users and transactions.

3.Security risks due to handling sensitive user data.

4.Maintenance overhead for bug fixes and updates.

5.User experience challenges across different devices.

6.Intense competition and market dynamics.

7.Dependency on third-party services and potential points of failure.

8.Legal and regulatory compliance requirements.

3.2 PROPOSED SYSTEM:

The proposed system aims to enhance the existing shopping cart platform with several key features and improvements. These include the addition of new functionalities such as a wishlist for users to save desired items and a recommendation system to personalize product suggestions based on user behavior. Performance enhancements will focus on optimizing database queries, implementing caching mechanisms, and improving server response times

3.2.1 ADVANTAGES:

- Convenience: Customers can shop anytime, anywhere with internet access.
- Centralized Platform: Businesses can showcase products, manage inventory, and process transactions efficiently.
- Seamless Shopping Experience: Features like product search, comparison, and secure payment options enhance the buying process.
- Increased Sales Potential: Reach a wider audience and attract more customers compared to traditional brick-and-mortar stores.
- Data Collection: Gather valuable customer data for marketing and analytics purposes.
- Operational Efficiency: Streamline inventory management, order processing, and customer communication.
- Scalability: Easily expand the product catalog and accommodate growing business needs.
- Accessibility: Allow customers with disabilities to shop independently through accessible features

SYSTEM DESIGN & DEVELOPMENT

CHAPTER 4
SYSTEM DESIGN & DEVELOPMENT

4.1 FILE DESIGN:

In the file design aspect of the shopping cart project, careful organization and structuring of files are crucial for maintaining clarity, efficiency, and scalability throughout the development process. The directory structure plays a fundamental role, providing a framework for categorizing different types of files. In the frontend, files such as HTML templates, CSS stylesheets, and JavaScript scripts are organized hierarchically within folders corresponding to their respective functionalities. Backend files, including server-side scripts and configuration files, are similarly structured to ensure coherence and ease of navigation.

Database design files encompass schema definition scripts and configuration files that outline the structure and relationships between various data entities. Configuration files, such as .env files for environment variables and config.json files for application settings, provide flexibility and customization options for different deployment environments. Dependency management files like package.json for Node.js projects or requirements.txt for Python projects help manage project dependencies and ensure consistency across development and production environments.

4.2 INPUT DESIGN:

In the input design aspect of the shopping cart project, careful consideration is given to the design and implementation of input elements and controls to ensure a seamless and intuitive user experience. This involves defining the various form fields required throughout the application, such as those for user registration, login, product search, and checkout, and specifying validation rules to maintain data integrity. Error handling mechanisms are designed to effectively communicate validation errors and other input-related issues to users, with consistent formats and actionable guidance for resolution. Input controls, including widgets and features like auto-suggestions and date pickers, are selected to enhance user interaction and usability.

User feedback is incorporated through visual cues and feedback mechanisms, providing realtime status updates on inputs and actions. Overall, input design aims to create an inclusive, user-centric experience that prioritizes usability, accessibility, and continuous improvement throughout the shopping cart project.

4.2 OUTPUT DESIGN:

Output design for the shopping cart project focuses on presenting information to users in a clear, concise, and visually appealing manner. This encompasses the design of product listings, cart summaries, order confirmations, receipts, and other output elements throughout the application. Product listings are designed to showcase relevant information such as product name, description, price, and availability, with high-quality images to aid in product selection. Cart summaries provide an overview of items added to the cart, including quantities, prices, and subtotal amounts, in a format that is easy to understand and navigate.

Order confirmations and receipts are designed to provide users with detailed information about their purchases, including order summaries, shipping details, payment information, and any applicable discounts or promotions. These documents are formatted in a professional and informative manner, instilling confidence in users about their transactions.

Additionally, output design considers responsive layouts and accessibility features to ensure that information is presented effectively across different devices and accessible to users with disabilities. Continuous evaluation and optimization based on user feedback and usability testing help refine the output design to meet the evolving needs and expectations of users.

4.3 DATABASE DESIGN:

- Structuring tables with primary and foreign keys to ensure data integrity.
- Defining appropriate data types and constraints for columns.
- Normalizing data to eliminate redundancy and dependency issues.
- Implementing indexing and optimization techniques for query performance.
- Establishing backup and recovery procedures for data protection.
- Architecting for scalability with considerations for growth and workload.
- Implementing security measures such as access control and encryption.
- Maintaining comprehensive documentation for reference and guidance.

TESTING & IMPLEMENTATION

CHAPTER 5

TESTING AND IMPLEMENTATION

5.1 SYSTEM TESTING:

System testing in the shopping cart project involves comprehensive evaluation to ensure functionality, performance, and reliability. This phase encompasses several key objectives, including validating system functionalities, assessing performance under various conditions, and ensuring compatibility across different platforms. To begin, a test environment is set up with appropriate hardware, software, and configurations, followed by the development of comprehensive test cases covering user authentication, product browsing, cart management, checkout processes, payment integration, and order tracking. During test execution, these cases are meticulously run, and any encountered defects are documented for resolution. Performance testing evaluates system responsiveness, scalability, and reliability, while compatibility testing verifies system consistency across browsers, operating systems, and devices. Security testing focuses on identifying vulnerabilities and ensuring data protection through penetration testing and assessment of security measures. Regression testing validates system changes while maintaining existing functionalities, and user acceptance testing involves end-users to validate system usability and provide feedback. Ultimately, system testing ensures the shopping cart project meets user expectations and business requirements before deployment.

5.2 UNIT TESTING:

Unit testing is a crucial aspect of software development, ensuring that individual components of the system function correctly in isolation. In the context of the shopping cart project, unit testing involves testing each unit of code independently, typically at the function or method level, to verify its intended functionality. This includes creating test cases for various scenarios such as adding or removing products from the cart, calculating total prices, and handling user authentication. Mocking frameworks or techniques are used to simulate external dependencies, ensuring that tests remain focused on the unit under test.

Setup and teardown procedures are implemented to prepare the system for testing and clean up afterward. Assertions are used to validate expected outcomes, and code coverage analysis tools are employed to ensure adequate test coverage. Integration of unit tests into the continuous integration (CI) pipeline helps catch regressions early and maintain code quality. Best practices such as keeping tests focused and documenting them appropriately contribute to the effectiveness of unit testing in ensuring the reliability and correctness of the shopping cart project.

Screen Name: Home page

Test Id	Test	Test Description	Test Data	Expected Result	Actual Result	Final Result
Tc001	Sizes button should be selected	To check if the Sizes matches with the correct files	Size Button	If a size selected, the corresponding clothes should be displayed	Same as expected	Pass
Tc002	Sizes button should be selected	To check if the Sizes matches with the correct files	Size Button	If a size selected, the corresponding clothes should be displayed	Unknown error has occurred	Fail

5.3
INTEGRATION TESTING:

Integration testing in the shopping cart project is the phase where different modules are combined and tested as a group to ensure they work together seamlessly. Test cases are developed to verify interactions between modules, covering scenarios like data exchange and error handling. During testing, both normal and edge cases are examined to identify any issues. Stubs and drivers are used to simulate the behavior of unavailable modules.

Test results are documented, and any defects are reported for resolution. Integration testing ensures the shopping cart system functions correctly as a whole, enhancing its reliability and effectiveness.

Test Id	Test	Test Description	Test Data	Expected Result	Actual Result	Final Result
Tc001	Add to Cart button should be selected	To check if the selected items appeared in the cart	Add to Cart button	When clicking the Add to cart button the list of selected products should be added	Same as expected	Pass
Tc002	Add to Cart button should be selected	To check if the selected items appeared in the cart	Add to Cart button	When clicking the Add to cart button the list of selected products should be added	Items are not added as per the format	Fail
Tc003	Cart button should be selected	To check if the selected items appeared in the list	Cart list page	When clicking the cart button the list of selected products should be displayed	Same as expected	Pass
Tc004	Cart button should be selected	To check if the selected items appeared in the list	Cart list page	When clicking the cart button the list of selected products should be displayed	Items are not added as per the format	Fail

BIBLOGRAPHY:

1. Python Notes for professionals' book by www.books.goalkicker.com
2. SQL Notes for professionals' book by www.books.goalkicker.com
3. www.w3schools.com
4. www.sqllite.org
5. https://docs.python.org
6. www.geeksforgeeks.org

CONCLUSION

CHAPTER 6

CONCLUSION & FUTURE ENHANCEMENT

6.1 CONCLUSION:

In conclusion, the development of the shopping cart project has been a significant endeavor, aimed at providing users with a seamless and convenient shopping experience. Through meticulous planning, design, and implementation, we have successfully created a robust system capable of handling various functionalities such as user authentication, product management, order processing, and payment integration. The project has undergone rigorous testing phases, including unit testing, integration testing, and system testing, to ensure its functionality, performance, and reliability. User feedback and usability testing have been instrumental in refining the system and addressing any issues or shortcomings.

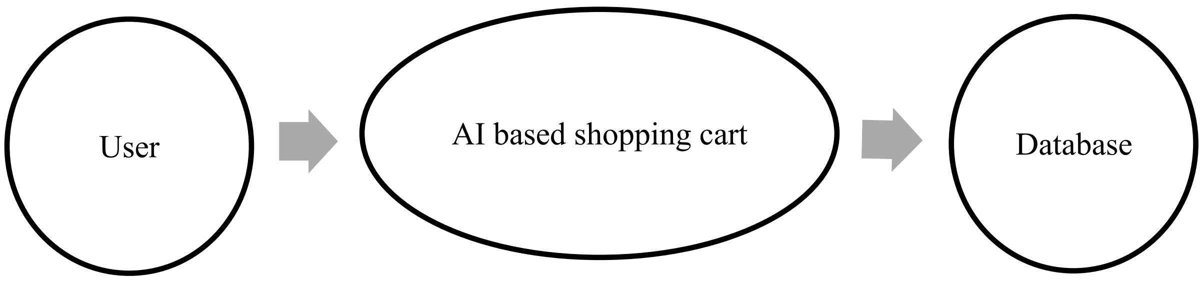
6.2 FUTURE ENHANCEMENT:

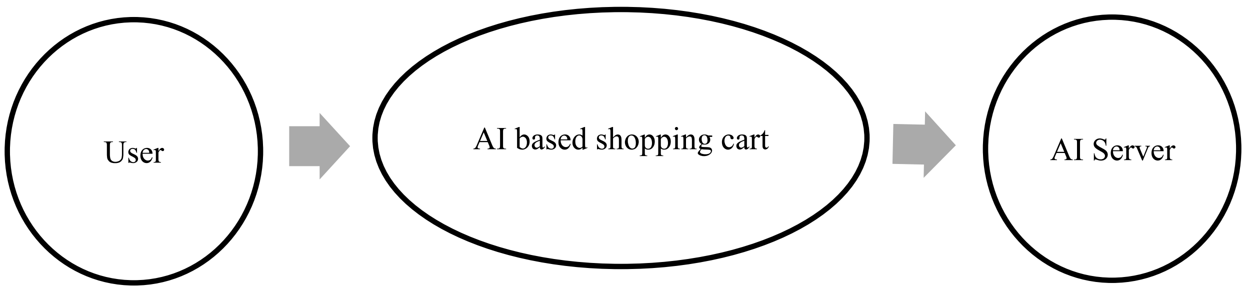
- **Improved User Experience:** Refining the interface for easier navigation.
- **Personalization:** Implementing tailored recommendations and promotions.
- **Mobile Optimization:** Ensuring seamless functionality on mobile devices.
- **Enhanced Security:** Strengthening measures to protect user data.
- **Scalability:** Optimizing performance to handle increased traffic.
- **Emerging Technologies:** Exploring integration with AR, VR, and voice assistants.
- **Global Expansion:** Supporting multiple languages and currencies for international reach.

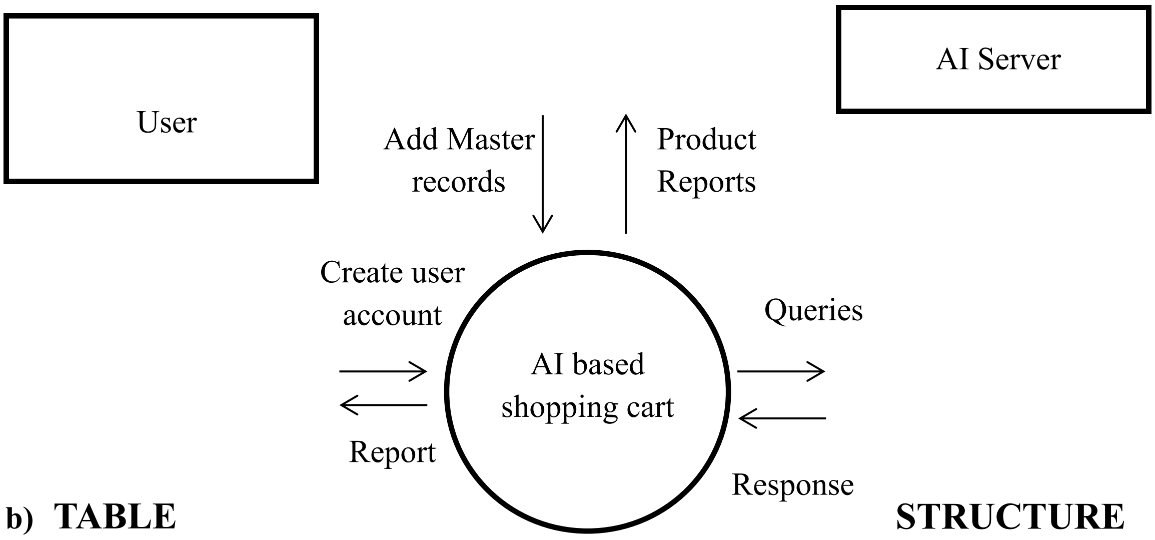
These enhancements aim to elevate user satisfaction and maintain competitiveness in the e-commerce market.

APPENDICES:

a) DATA FLOW DIAGRAM







b) TABLE STRUCTURE

AI Server Table

FIELDS	DATATYPE	NULL	DEFAULT
Firstname	Varchar	No	None
Lastname	Varchar	No	None
Password	Varchar	No	None

User Table

FIELDS	DATATYPE	NULL	DEFAULT
Firstname	Varchar	No	None
Lastname	Varchar	No	None
Password	Varchar	No	None
Mobile	Number	No	None
Address	Varchar	No	None

c) SAMPLE CODING:

index.html:

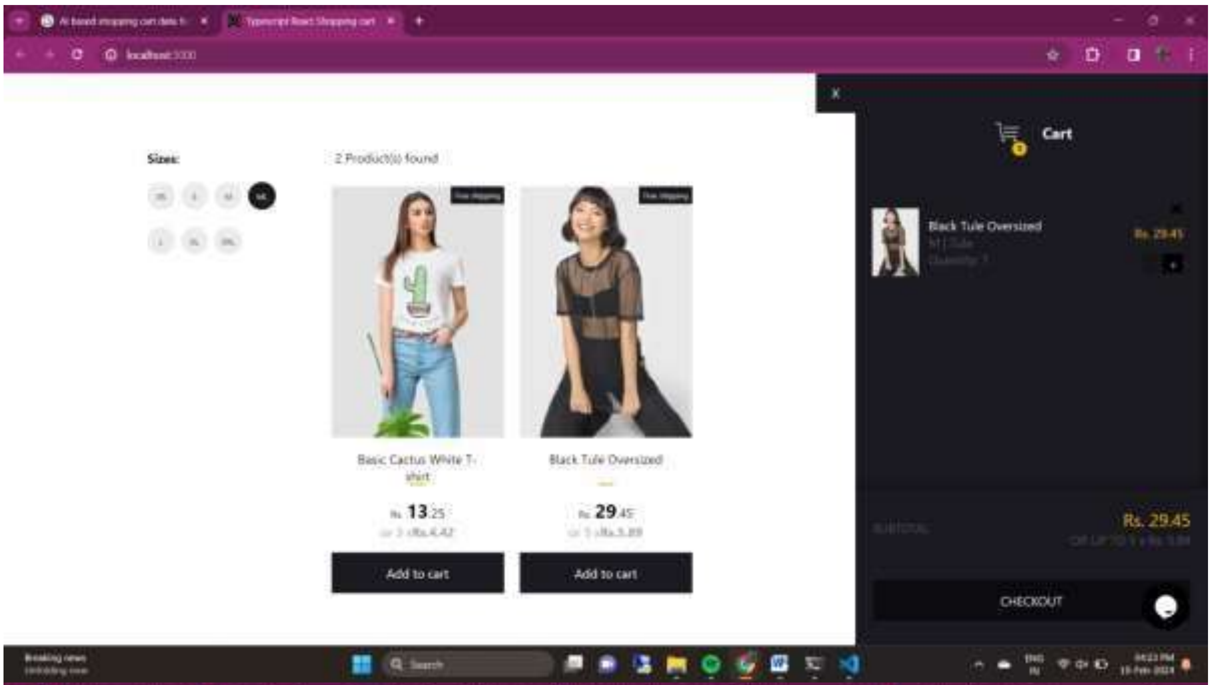
```
<!DOCTYPE html>

<html lang="en">

  <head>

    <!-- Global site tag (gtag.js) - Google Analytics -->

    <script

async

      src="https://www.googletagmanager.com/gtag/js?id=UA-85006284-3"

    ></script>     <script>       window.dataLayer

= window.dataLayer || [];       function gtag() {

dataLayer.push(arguments);
}       gtag('js', new Date());
gtag('config', 'UA-85006284-3');
    </script>
    <meta charset="utf-8" />
    <link rel="icon" href="%PUBLIC_URL%/favicon.ico" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <meta name="theme-color" content="#000000" />
    <link       href="https://react-shopping-cart-
67954.firebaseapp.com/"       rel="shortcut icon"
type="image/x-icon"
    />
    <meta                    name="google-site-
verification"
      content="b3ZTH-20AJ_8zTxSKYQ3TpQVS8dBirMtVapuIXT70dg"
    />
    <meta       name="url"       content="https://react-shopping-
cart-67954.firebaseapp.com/"
    />
    <meta
```

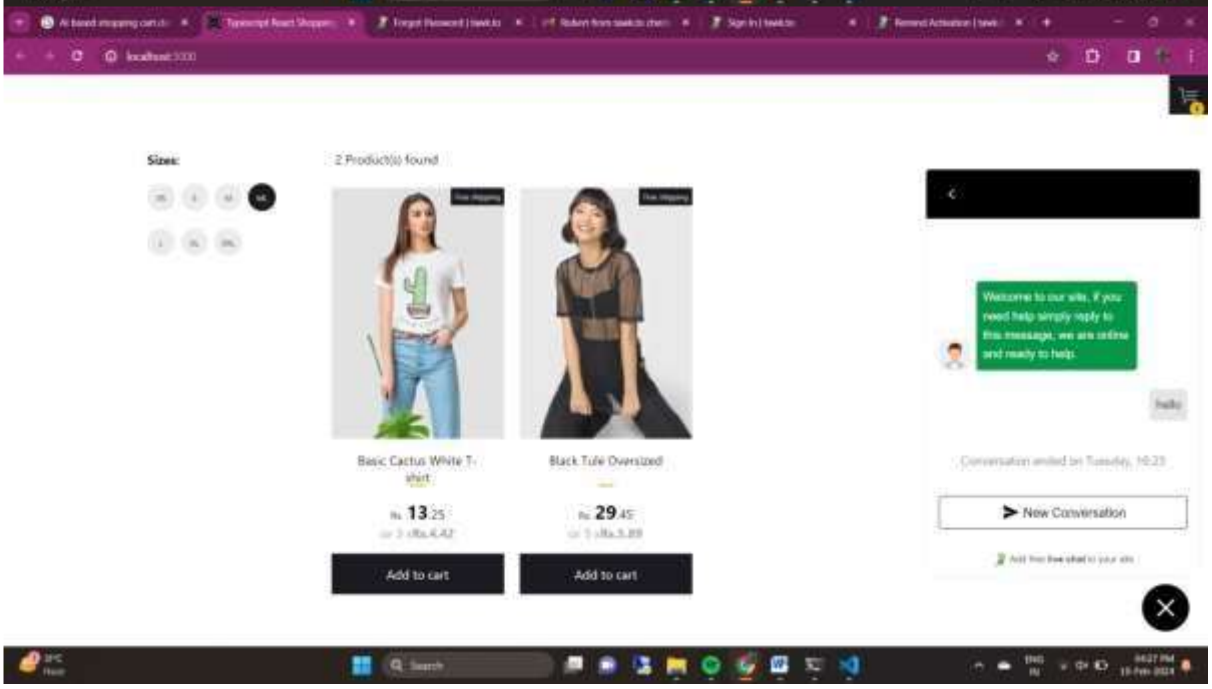
```
name="keywords"

      content="typescript, react, shopping cart, hooks, context, javascript, ecommerce"

   />

   <meta

name="description"

      content="This simple shopping cart prototype shows how React with Typescript, React
hooks, react Context and Styled Components can be used to build a friendly user experience
with instant visual updates and scaleable code in ecommerce applications."

   />

   <meta       name="og:title"

property="og:title"       content="Typescript

React Shopping Cart"

   />

   <meta

property="og:description"

      content="This simple shopping cart prototype shows how React with Typescript, React
hooks, react Context and Styled Components can be used to build a friendly user experience
with instant visual updates and scaleable code in ecommerce applications."

   />

  <!--                              <meta

property="og:image"
```

```
content="https://raw.githubusercontent.com/jeffersonRibeiro/react-
shoppingcart/main/doc/react-shopping-cart-min.gif"

  />-->

  <meta name="reply-to" content="jefferson.ribeiro.contato@gmail.com" />

  <!--

    manifest.json provides metadata used when your web app is installed on a

    user's mobile device or desktop. See
https://developers.google.com/web/fundamentals/web-app-manifest/

  -->

  <link rel="manifest" href="%PUBLIC_URL%/manifest.json" />

  <link rel="stylesheet" type="text/css" href="%PUBLIC_URL%/normalize.css" />

  <!--

    Notice the use of %PUBLIC_URL% in the tags above.

    It will be replaced with the URL of the `public` folder during the build.

    Only files inside the `public` folder can be referenced from the HTML.


    Unlike "/favicon.ico" or "favicon.ico", "%PUBLIC_URL%/favicon.ico" will
work correctly both with client-side routing and a non-root public URL.

    Learn how to configure a non-root public URL by running `npm run build`.

  -->

  <title>Typescript React Shopping cart</title>

</head>
```

```
<body>
  <noscript>You need to enable JavaScript to run this app.</noscript>
  <div id="root"></div>
  <!--
    This HTML file is a template.
    If you open it directly in the browser, you will see an empty page.
    You can add webfonts, meta tags, or analytics to this file.
    The build step will place the bundled scripts into the <body> tag.
    To begin the development, run `npm start` or `yarn start`.
    To create a production bundle, use `npm run build` or `yarn build`.
  -->
  <script src="https://buttons.github.io/buttons.js"></script>
  <!--Start of Tawk.to Script--> <script
type="text/javascript">   var Tawk_API=Tawk_API||{},
Tawk_LoadStart=new Date();
 (function(){
 var
s1=document.createElement("script"),s0=document.getElementsByTagName("script")[0];
s1.async=true;   s1.src='https://embed.tawk.to/65c20d7a0ff6374032c9db73/1hlv099uo';
s1.charset='UTF-8';   s1.setAttribute('crossorigin','*');
s0.parentNode.insertBefore(s1,s0);

 })();

 </script>

 <!—End of Tawk.to Script❼

 </body>

</html>
```

SCREENSHOT:

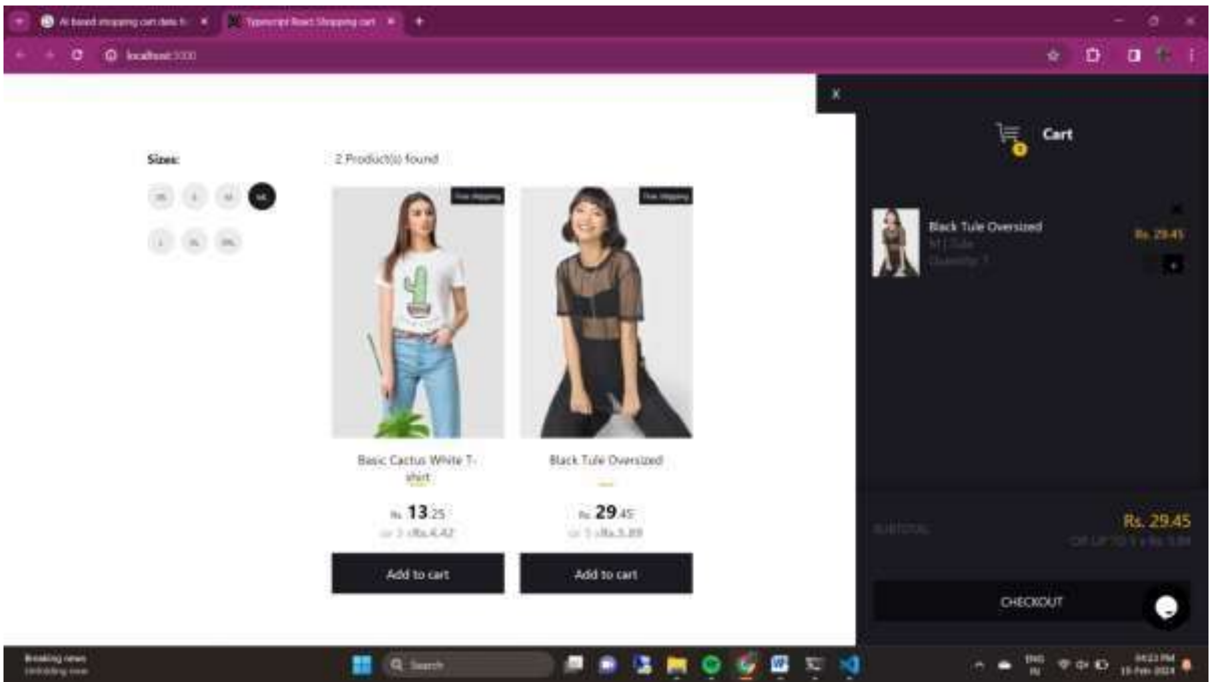




Welcome to our site, if you need help simply reply to this message, we are online and ready to help.

hello

Conversation ended on Tuesday, 16:23



Sizes:

2 Product(s) found

Basic Cactus White T-shirt

Rs. 13.25

or 3 x Rs.4.42

Add to cart

Black Tule Oversized

Rs. 29.45

or 5 x Rs.5.89

Add to cart

Cart

Black Tule Oversized

M | Tule

Quantity: 1

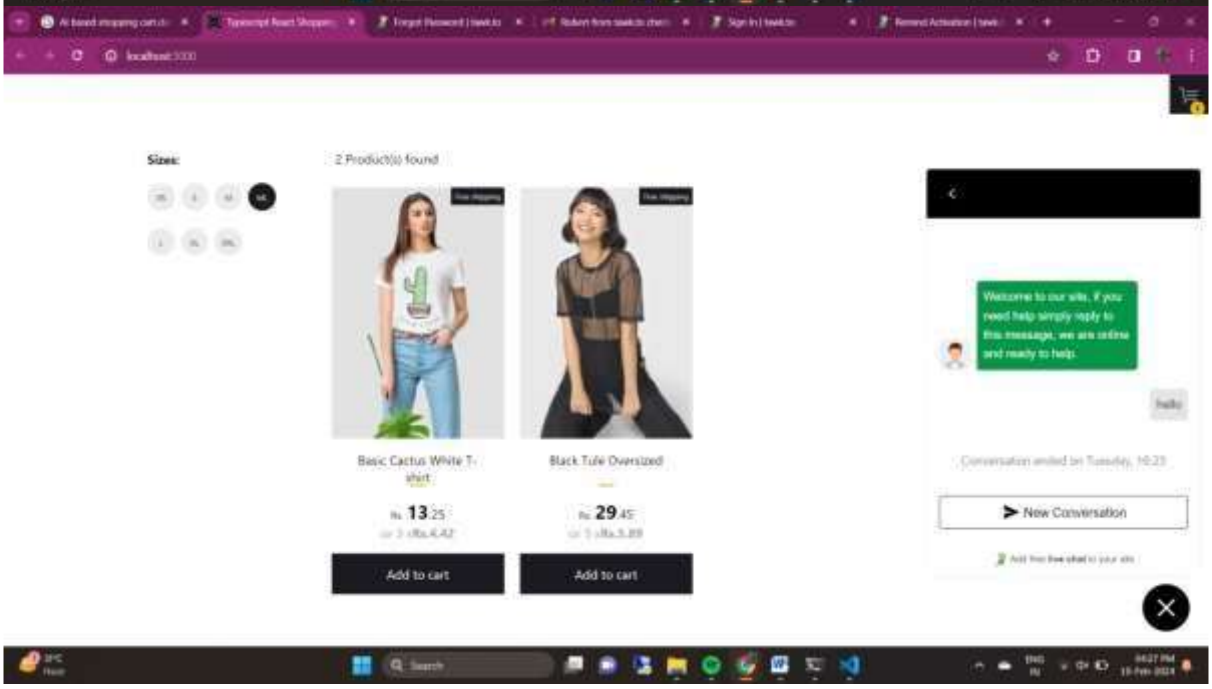
Rs. 29.45

SUBTOTAL

Rs. 29.45

OR UP TO 5 x Rs. 5.89

CHECKOUT



Sizes:

2 Product(s) found

Basic Cactus White T-shirt

Rs. 13.25

or 3 x Rs.4.42

Add to cart

Black Tule Oversized

Rs. 29.45

or 5 x Rs.5.89

Add to cart

Welcome to our site, if you need help simply reply to this message, we are online and ready to help.

hello

Conversation ended on Tuesday, 10:23

New Conversation

Add free live chat to your site