

MERN Stack developer

Programming Pathshala

A training report

Submitted in partial fulfillment of the requirements for the award of degree of

BTech (Full Stack Development)

Submitted to

LOVELY PROFESSIONAL

UNIVERSITY PHAGWARA, PUNJAB



SUBMITTED BY

Name of student:

Ritik Kumar

Registration Number:

12200510

**Signature of the
student:**

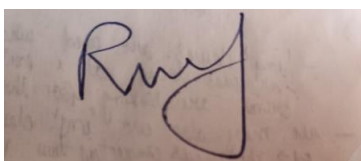


Table Of Contents

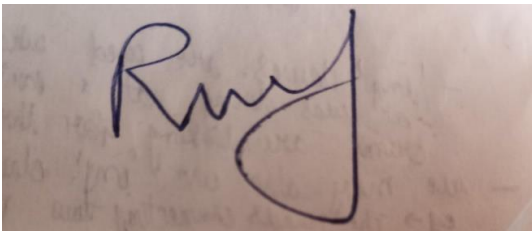
- **Declaration**
- **Acknowledgement**
- **Certification**
- **TECHNOLOGY LEARNT**
- **List Of Abbreviations**
- **Project: Deploy a Static Website on Payment Gateway Development**
- **Steps to Deploy a Static web site with screenshot:**
- **Conclusion**
- **Reference**

Declaration

To whom so ever it may concern

I, **Ritik Kumar, 12200510**, hereby declare that the work done by me on “**MERN STACK**” from **June, 2024** to **September, 2024**, is a record of original work for the partial fulfillment of the requirements for the award of the degree, **Bachelor of Computer Science.**

Ritik Kumar (12200510)

A photograph of a handwritten signature in blue ink on a light-colored surface. The signature is stylized, starting with a large 'R' and ending with a long, sweeping horizontal stroke.

Dated: 29th August, 2024

CERTIFICATE

Summer Training Program

Ritik Kumar

has successfully completed course on

MERN Stack Development with a Capstone Project

A Program offered by Programming Pathshala, to be excellent Software Engineers.



2024-07-20

DATE



programmingpathshala.com



Programming
Pathshala

Anoop Garg

Co-founder, Programming Pathshala



Programming Pathshala
669ba48775386515a8255998

Acknowledgement

I would like to express my gratitude towards my university as well as Programming Pathshala for providing me the golden opportunity to do this wonderful summer training regarding MERN STACK WEB DEVELOPMENT, which also helped me in doing a lot of homework and learning. As a result, I came to know about so many new things. So, I am thankful to them.

Moreover, I would like to thank my friends who helped me a lot whenever I got stuck in some problem related to my course. I am thankful to have such a good support from them as they always have my back whenever I need it.

I have made efforts in this project. However, it would not have been possible without the kind support and help of many individuals and organizations. I would like to extend my sincere thanks to all of them.

Deepest thanks to our Trainer Likhilesh Balpande (Mentor) for their guidance, monitoring, constant encouragement and correcting various assignments of ours with attention and care. They have taken the pain to go through the project and training sessions and make necessary corrections as needed and we are very grateful for that.

Sincerely,
RITIK KUMAR.

Technology Learnt

1. **Express.js:**

- Server-side framework for Node.js.
- Middleware functions for handling HTTP requests and responses.
- RESTful API development.
- Route management and handling.

2. **React.js:**

- Front-end JavaScript library for building user interfaces.
- Component-based architecture.
- JSX (JavaScript XML) syntax.
- State management with Hooks (use State, use Effect, etc.).
- React Router for SPA (Single Page Application) navigation.

3. **Node.js:**

- JavaScript runtime environment for server-side scripting.
- Asynchronous programming with callbacks, Promises, and async/await.
- NPM (Node Package Manager) for managing dependencies.
- Building and managing a backend server.

4. **JavaScript (ES6+):**

- Modern JavaScript syntax and features (e.g., arrow functions, destructuring, spread/rest operators).
- Object-oriented programming concepts in JavaScript.
- Event-driven programming.

5. **RESTful APIs:**

- Designing and implementing RESTful services.
- JSON data format for API communication.
- Handling HTTP methods (GET, POST, PUT, DELETE).

6. **Version Control (Git & GitHub):**

- Source code management using Git.
- Collaborating on projects using GitHub.
- Branching, merging, and pull requests.

7. MongoDB:

- NoSQL database management.
- Document-oriented data storage.
- CRUD operations (Create, Read, Update, Delete).
- Aggregation framework for data analysis.

8. Postman:

- API testing and debugging.
- Sending HTTP requests and inspecting responses.
- Automation of API tests.

9. Webpack/Babel:

- Module bundling and asset management in JavaScript applications.
- Transpiling modern JavaScript (ES6+) to browser-compatible versions.

10. Authentication & Security:

- Implementing user authentication using JWT (JSON Web Tokens).
- Securing routes and endpoints.
- Password hashing and user session management.

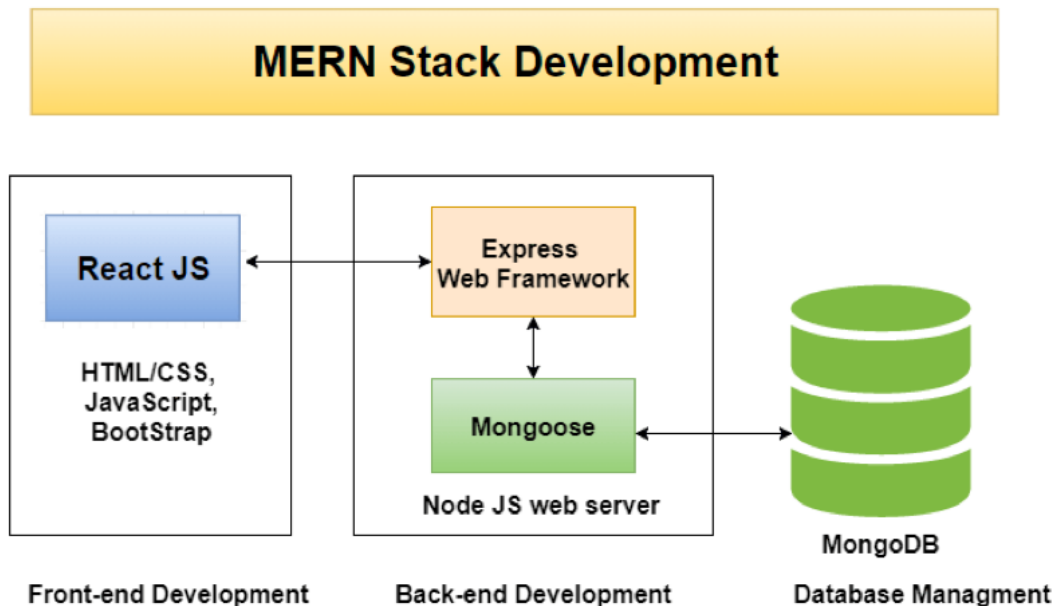
11. Deployment:

- Deploying MERN stack applications on cloud platforms (e.g., Heroku, AWS).
- Setting up CI/CD pipelines.
- Managing environment variables and configurations for production.

List of Abbreviation

- MERN: MongoDB, Express.js, React.js, Node.js
- API: Application Programming Interface
- JSON: JavaScript Object Notation
- MVC: Model-View-Controller
- REST: Representational State Transfer
- CRUD: Create, Read, Update, Delete.
- HTTP: Hyper Text Transfer Protocol
- UI: User Interface
- DOM: Document Object Model
- JSX: JavaScript XML (used in React)
- SPA: Single Page Application
- MVC: Model-View-Controller
- NoSQL: Non-Structured Query Language (related to MongoDB)

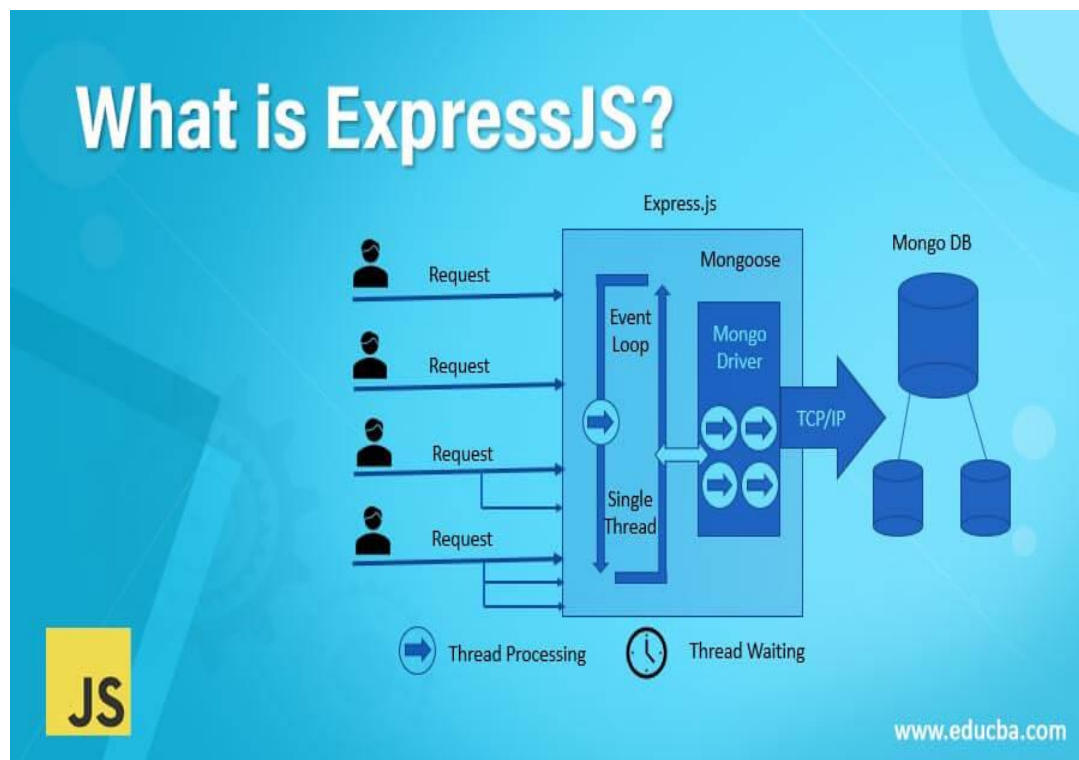
MERN :- MERN is a popular technology stack used for building web applications. It consists of four main technologies:



1. **MongoDB:** A NoSQL database that stores data in a flexible, JSON-like format. It allows for easy scalability and is well-suited for applications that require a lot of data.



2. **Express.js:** A web application framework for Node.js that simplifies the process of building server-side applications. It provides robust features for web and mobile applications, including routing, middleware support, and more.



3. **React:** A JavaScript library for building user interfaces, particularly single-page applications (SPAs). Developed by Facebook, React allows developers to create reusable UI components and manage the state of applications efficiently.

4. **Node.js:** A JavaScript runtime built on Chrome's V8 engine that allows developers to execute JavaScript code on the server side. It enables the creation of scalable network applications and is particularly well-suited for I/O-heavy tasks.



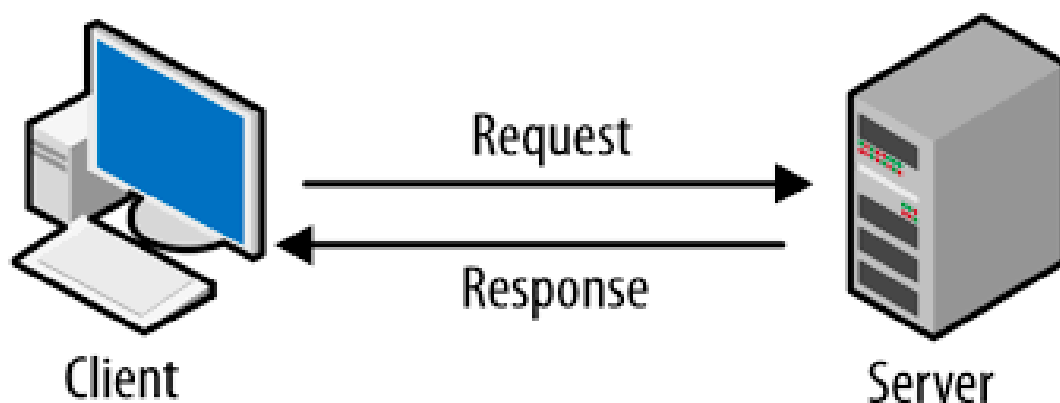
Key Features of MERN

- **Full-Stack Development:** MERN enables developers to work on both the front-end and back-end using JavaScript, making it easier to manage and unify the codebase.
- **JSON Data Format:** The stack uses JSON for data interchange, which simplifies the process of sending and receiving data between the client and server.
- **Scalability:** Each component of the MERN stack can be scaled independently, allowing applications to grow as needed.
- **Rich Ecosystem:** Each technology in the MERN stack has a large community and a wealth of libraries and tools that can enhance development.

Typical Workflow

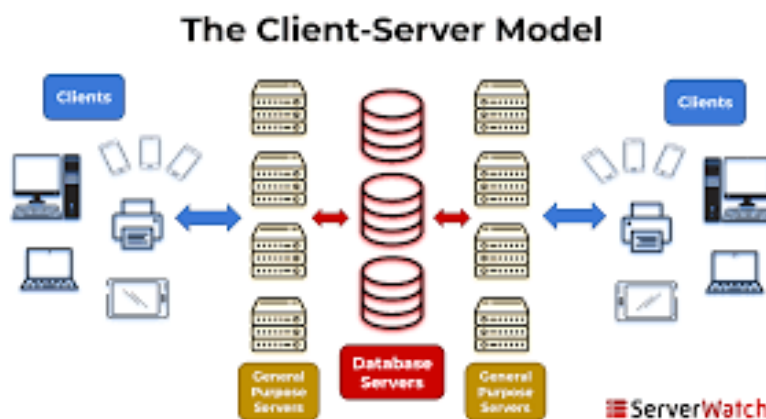
1. Client-Side (React Front-End)

- **User Interaction:** The front-end is built using React, which allows developers to create interactive user interfaces. Users interact with various components (like buttons, forms, etc.) that trigger events.
- **Component-Based Architecture:** React uses a component-based architecture, meaning the UI is broken down into reusable components. Each component can manage its own state and lifecycle, making it easier to build complex UIs.
- **State Management:** React applications often use state management libraries (like Redux or Context API) to manage the application's state. This ensures that when data changes, the UI updates accordingly.
- **API Requests:** When the user performs an action that requires data from the server (like submitting a form), the React app sends an HTTP request (usually using libraries like Axios or Fetch) to the Express server. This request typically includes the necessary data in the body or as query parameters.



2. Server-Side (Express.js)

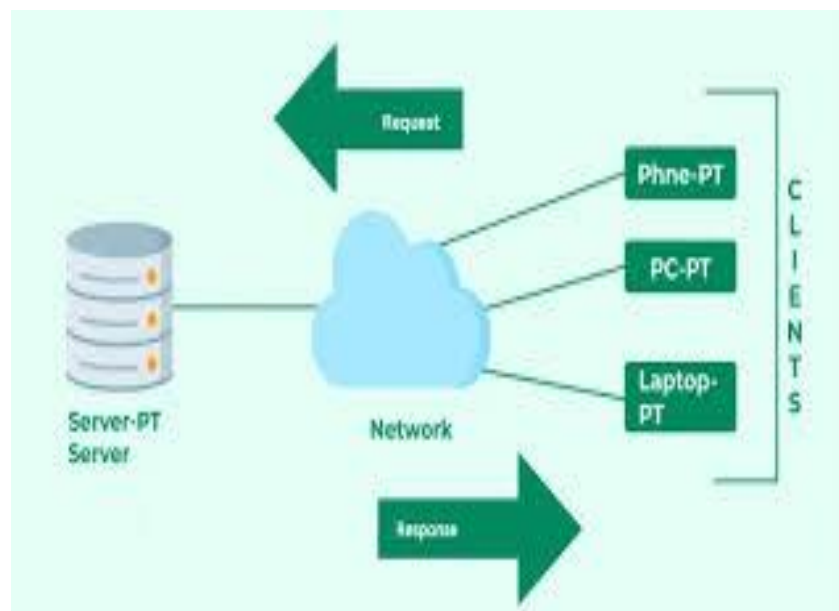
- **Routing:** Express.js acts as a web server and defines routes that correspond to different API endpoints. Each route can handle different HTTP methods (GET, POST, PUT, DELETE) based on the type of operation needed.
- **Middleware:** Express uses middleware functions to process incoming requests. Middleware can perform tasks such as logging requests, parsing request bodies (using libraries like body-parser), and handling authentication.
- **Request Handling:** When an API request is received, Express routes it to the appropriate handler function. This function will process the request, perform any necessary logic, and interact with MongoDB if needed.



- **Database Interaction:** Express communicates with MongoDB using a library like Mongoose, which provides a straightforward way to interact with the database. It allows developers to define schemas and models, making it easier to validate data.

3. Database (MongoDB)

- **Data Storage:** MongoDB is a NoSQL database that stores data in collections of documents. Each document is a JSON-like object, which allows for flexible data structures.
- **CRUD Operations:** The server can perform CRUD (Create, Read, Update, Delete) operations on the data stored in MongoDB. For example:
 - **Create:** When a user submits a new form, the server can create a new document in the database.
 - **Read:** The server can retrieve data from the database to send back to the client, such as user profiles or product listings.
 - **Update:** When a user edits their information, the server updates the corresponding document in the database.
 - **Delete:** The server can remove documents from the database when necessary.



Project: Deploy a Static Website on Payment Gateway Development

Introduction to the Test Website on Payment Gateway Integration:-

In today's digital economy, secure and efficient payment processing is crucial for online businesses. To explore and demonstrate the implementation of payment processing within web applications, a test website was developed to integrate a payment gateway. This project aims to provide hands-on experience with the technical aspects of payment gateway integration, emphasizing security, user experience, and transaction management.

The test website is designed as a mock e-commerce platform where users can simulate purchasing products or services. The primary objective is to enable seamless transactions, ensuring that user data is handled securely, and that the payment process is intuitive and reliable. Key features include user authentication, shopping cart management, and real-time payment processing.

By utilizing a popular payment gateway (e.g., Stripe, PayPal, or Razor pay), the website demonstrates the full payment cycle—from selecting products and entering payment details to receiving transaction confirmations. This project not only enhances understanding of backend processes like API integration and database management but also highlights best practices in handling sensitive information and complying with payment industry standards.

1. Define the Project Scope

- **Purpose of the Website:** Determine the primary objectives of the test website, such as simulating online transactions, demonstrating payment gateway integration, and testing various payment scenarios.
- **Identify Required Features:** List essential features like user authentication, product selection, shopping cart, payment processing, and order confirmation.

2. Choose a Payment Gateway

- **Research Payment Gateways:** Evaluate popular payment gateways (e.g., Stripe, PayPal, Razor pay) based on factors like ease of integration, transaction fees, supported payment methods, and documentation. I use Stripe in my website as gateway.
- **Set Up a Developer Account:** Register for a developer account with the chosen payment gateway to gain access to sandbox environments, API keys, and SDKs.

3. Set Up the Development Environment

- **Install Required Tools:** Set up your local development environment with tools like a text editor (e.g., Visual Studio Code), version control (Git), and a local server.
- **Create a New Project Directory:** Organize the project files into directories for HTML, CSS, JavaScript, and any other resources.

4. Design the Website

- **Create the Front-End Interface:** Design the user interface using HTML and CSS. The design should include:

- Product Display: A section for displaying Dishes or services available for purchase.
- Dishes Cart: A simple dishes cart where users can add or remove items.
- Checkout Page: A checkout form to collect user information, shipping details, and payment information.
- Environment: I use React to give environment to my website and give different packages in my server and add client for store the project material.
- Implement Responsive Design: Use CSS frameworks like Bootstrap or Tailwind CSS to ensure the website is responsive across different devices.

5. Integrate the Payment Gateway

- Embed Payment Gateway SDK: Integrate the payment gateway's client-side SDK (e.g., Stripe.js, PayPal Buttons) into your website. This allows you to securely handle payment details on the frontend.
- Configure API Keys: Use the test API keys provided by the payment gateway to set up the sandbox environment. This will enable you to simulate transactions without involving real money.
- Develop Payment Logic: Write JavaScript code to handle payment submission, validate user inputs, and process payments using the gateway's API.
- Tokenization: Implement tokenization to securely transmit payment details without storing sensitive information on your servers.

6. Back-End Development (Optional)

- Set Up a Server: If the project requires back-end processing (e.g., for

order management or database storage), set up a server using Node.js, Express, or any other server-side technology.

- **Create API Endpoints:** Develop endpoints to handle payment confirmations, order details, and user data. These endpoints will interact with the payment gateway's API to verify and store transaction data.
- **Packages:** Connect the back-end to a different packages such as Dotenv ,Nodemon, Stripe, cors, Express and pkg.json in my project to give the dependencies to run the website .

7. Test the Payment Process

- **Perform End-to-End Testing:** Use the payment gateway's stripe environment to simulate transactions from start to finish, ensuring that each step (product selection, payment processing, order confirmation) works as expected.
- **Handle Edge Cases:** Test various scenarios such as invalid card details, expired cards, and insufficient funds to ensure the website handles errors gracefully.
- **Validate Security Measures:** Ensure that all security protocols are in place, such as HTTPS, input validation, and CSRF protection.

8. Deploy the Test Website

- **Choose a Hosting Platform:** Deploy the website to a hosting platform (e.g., GitHub Pages, Netlify, Heroku) that supports static or dynamic sites, depending on your website's requirements.
- **Set Up Domain and SSL:** If desired, configure a custom domain and secure the site with an SSL certificate to ensure encrypted transactions.
- **Final Testing:** After deployment, perform a final round of testing in

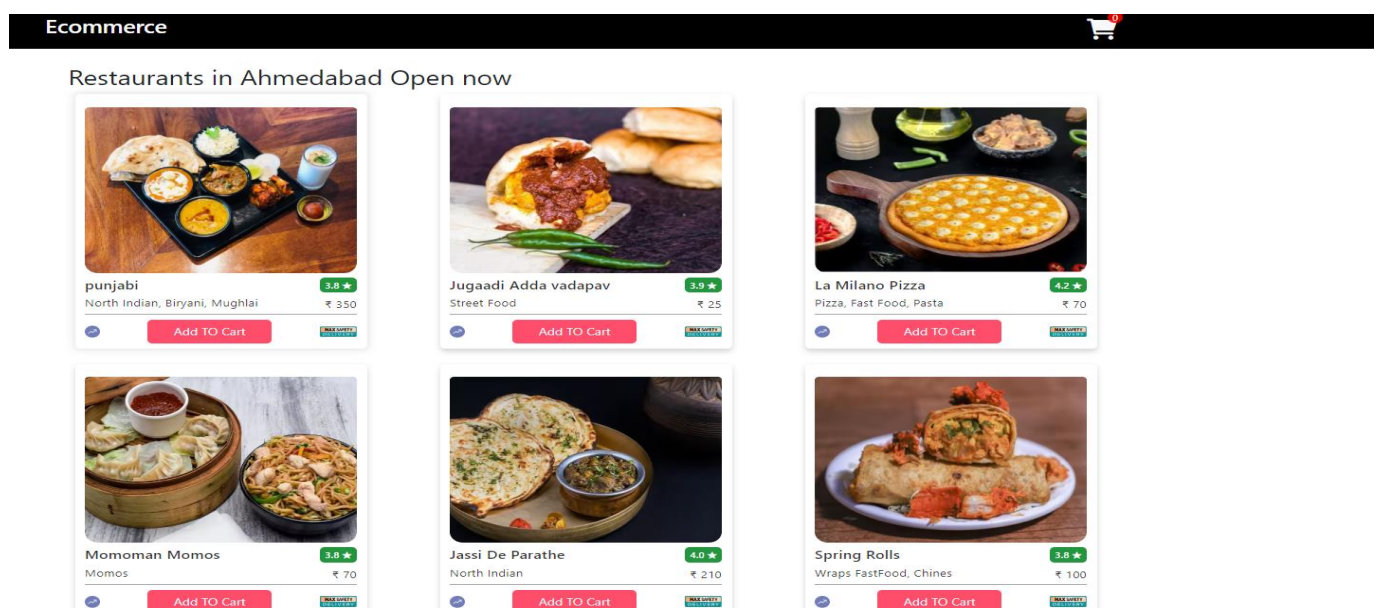
the live environment to confirm that all features work as expected.

9. Monitor and Iterate

- **Collect Feedback:** Gather feedback from test users on the payment process and user experience.
- **Make Improvements:** Based on feedback and test results, refine the user interface, improve performance, and enhance security measures.
- **Update Documentation:** Maintain detailed documentation of the integration process, including API references, configuration settings, and testing results for future reference.

Steps to Deploy a Static web site with screenshot:

Step 1: Frontend Page (UI of Dishes): Created a visually appealing webpage that showcases various dishes with images, descriptions, and prices. Users can browse through the menu and select their desired items.



Step 2: Add to Cart and Checkout Page: Implement a shopping cart feature where users can view their selected dishes, adjust quantities, and proceed to checkout.

Cart Calculation(2)

EmptyCart

Action	Product	Name	Price	Qty			Total Amount
<div></div>	<div></div>	Jugaadi Adda vadapav	25	<div>-</div>	<div>1</div>	<div>+</div>	₹ 25
<div></div>	<div></div>	La Milano Pizza	70	<div>-</div>	<div>1</div>	<div>+</div>	₹ 70
Items In Cart : 2			Total Price : ₹ 95			Checkout	

Step 3: Click on Pay Button To pay: Choose any options such as Card, Net banking, Wallet and Pay later to pay. Then Click Continue to proceed the payment.



Pay

₹95.00



Jugaadi Adda vadapav

₹25.00



La Milano Pizza

₹70.00

Pay with card

Email

r@lpu.in

Card information

4000 0035 6000 0008

VISA

01 / 25

123



Cardholder name

Rahul

Country or region

India



☒ Securely save my information for 1-click checkout

Pay faster on this site and everywhere Link is accepted.

89551 59068

By saving my info, I agree to the Link [Terms](#) and [Privacy Policy](#).

link

Step 4: If you want to check the Payment History then go stripe site then go Customer section then check the payment history.

Test mode

You're using test data. To accept payments, complete your business profile.

Complete pro

New Business

Home

Balances

Transactions

Customers

Product catalog

Products

Payments

Billing

Reporting

More

Q Search

Developers

Test mode

Payments

Amount	Description	Date
₹95.00 INR	Succeeded ✓ pi_3PtmCQRoHjMuu03R0Fi5a04e	Aug 31, 1:28 PM

1 result

Payment methods

>

VISA

Visa •••• 0008

Expires Jan 2025

Recent activity

Insights

Spent

\$1.13

Details

Guest details

Rahul

r@lpu.in

Customer since

Aug 31

Billing details

India

Conclusion

In conclusion, this report has detailed the development and deployment process of a test website focused on integrating a payment gateway using the MERN stack. The project involved a comprehensive exploration of front-end and back-end technologies, highlighting the importance of secure and efficient payment processing in modern web applications. Through the steps outlined, the test website successfully demonstrated the full payment cycle—from user authentication and product selection to payment processing and order confirmation. This exercise not only provided valuable hands-on experience with the technical aspects of payment gateway integration but also underscored best practices in security and user experience design.

By leveraging tools such as Express.js, React.js, and Node.js, and integrating a popular payment gateway, the project exemplified the practical application of the MERN stack in creating a robust and scalable web solution. The deployment process further emphasized the significance of testing and monitoring to ensure the reliability and security of online transactions. Overall, this project serves as a strong foundation for future work in e-commerce development and payment system integration, equipping developers with the skills and knowledge necessary to build secure and efficient online payment solutions.

Reference

- Express.js Guide. (n.d.). Retrieved from <https://expressjs.com/en/guide/routing.html>
- React.js Documentation. (n.d.). Retrieved from <https://reactjs.org/docs/getting-started.html>
- Node.js v16.x Documentation. (n.d.). Retrieved from <https://nodejs.org/en/docs/>
- Stripe Integration Guide. (n.d.). Retrieved from <https://stripe.com/docs/integration/quickstart>
- PayPal Developer Documentation. (n.d.). Retrieved from <https://developer.paypal.com/docs/api/overview/>
- GitHub Pages Documentation. (n.d.). Retrieved from <https://docs.github.com/en/pages>
- Netlify Documentation. (n.d.). Retrieved from <https://docs.netlify.com>