**Lesson 07 Demo 06**

**Creating and Executing Scenario-Based Tests Using   
Generative AI**

**Objective:** To create and execute comprehensive scenario-based test scripts using Generative AI tools, focusing on real-world user interactions within a payment gateway system to enhance efficiency and accuracy in software development

**Tools required:** VS Code, Microsoft Copilot, Node JS and JEST testing framework

**Prerequisites:** Basic knowledge and understanding of scenario-based testing and writing test scripts using Selenium

Steps to be followed:

1. Generate the test script using Microsoft Copilot
2. Execute the test script using Eclipse IDE and Selenium

|  |
| --- |
| **Note:** Please note that all the generative AI tools used in this exercise can produce varied outputs even when presented with similar prompts. Thus, you may get different outputs for the same prompt. |

**Step 1:** **Generate the test script using Microsoft Copilot**

* 1. Open VS Code and create an HTML file named **index.html**

A screenshot of a computer

Description automatically generated

1. Add the following code to the **index.html** file:

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

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

    <title>Payment Gateway</title>

    <style>

        body {

            font-family: Arial, sans-serif;

            margin: 0;

            padding: 0;

            background-color: #f7f7f7;

            display: flex;

            justify-content: center;

            align-items: center;

            height: 100vh;

        }

        .container {

            background-color: #ffffff;

            padding: 40px;

            border-radius: 8px;

            box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);

            width: 400px;

            max-width: 100%;

            text-align: center;

        }

        h1 {

            margin-bottom: 20px;

            color: #333333;

        }

label {

            display: block;

            margin-bottom: 5px;

            color: #555555;

            text-align: left;

        }

        input[type="text"],

        input[type="number"],

        select {

            width: calc(100% - 22px);

            padding: 10px;

            border: 1px solid #cccccc;

            border-radius: 4px;

            box-sizing: border-box;

            margin-bottom: 15px;

        }

        button {

            width: 100%;

            padding: 12px;

            background-color: #007bff;

            color: #ffffff;

            border: none;

            border-radius: 4px;

            cursor: pointer;

            font-size: 16px;

            transition: background-color 0.3s;

        }

        button:hover {

            background-color: #0056b3;

        }

        .error-message {

            color: #ff0000;

            margin-top: 5px;

            text-align: left;

        }

    </style>

</head>

<body>

    <div class="container">

        <h1>Secure Payment Gateway</h1>

        <form id="paymentForm" onsubmit="return validatePayment()">

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

            <input type="text" id="cardNumber" placeholder="Enter your card number" required>

            <div class="error-message" id="cardNumberError"></div>

            <label for="cardHolderName">Cardholder Name:</label>

            <input type="text" id="cardHolderName" placeholder="Enter cardholder name" required>

            <div class="error-message" id="cardHolderNameError"></div>

            <label for="expiryDate">Expiry Date:</label>

            <input type="text" id="expiryDate" placeholder="MM/YYYY" required>

            <div class="error-message" id="expiryDateError"></div>

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

            <input type="text" id="cvv" placeholder="Enter CVV" required>

            <div class="error-message" id="cvvError"></div>

            <label for="amount">Amount:</label>

            <input type="number" id="amount" placeholder="Enter amount" required>

            <div class="error-message" id="amountError"></div>

            <label for="currency">Currency:</label>

            <select id="currency" required>

                <option value="USD">USD</option>

                <option value="EUR">EUR</option>

                <option value="GBP">GBP</option>

            </select>

            <button type="submit">Pay Now</button>

        </form>

    </div>

<script src="paymentValidate.js"></script>

</body>

</html>

**paymentValidate.js** code

function validatePayment() {

    var cardNumber = document.getElementById('cardNumber').value;

    var cardHolderName = document.getElementById('cardHolderName').value;

    var expiryDate = document.getElementById('expiryDate').value;

    var cvv = document.getElementById('cvv').value;

    var amount = document.getElementById('amount').value;

    var cardNumberError = document.getElementById('cardNumberError');

    var cardHolderNameError = document.getElementById('cardHolderNameError');

    var expiryDateError = document.getElementById('expiryDateError');

    var cvvError = document.getElementById('cvvError');

    var amountError = document.getElementById('amountError');

    // Simple validation for demonstration purposes

    var isValid = true;

    if (!/^(\d{16})$/.test(cardNumber)) {

        cardNumberError.textContent = "Please enter a valid card number.";

        isValid = false;

    } else {

        cardNumberError.textContent = "";

    }

    if (cardHolderName.trim() === "") {

        cardHolderNameError.textContent = "Please enter the cardholder name.";

        isValid = false;

    } else {

        cardHolderNameError.textContent = "";

    }

    if (!/^(0[1-9]|1[0-2])\/\d{4}$/.test(expiryDate)) {

        expiryDateError.textContent = "Please enter a valid expiry date in MM/YYYY format.";

        isValid = false;

    } else {

        expiryDateError.textContent = "";

    }

    if (!/^\d{3}$/.test(cvv)) {

        cvvError.textContent = "Please enter a valid CVV.";

        isValid = false;

    } else {

        cvvError.textContent = "";

    }

    if (isNaN(amount) || amount <= 0) {

        amountError.textContent = "Please enter a valid amount.";

        isValid = false;

    } else {

        amountError.textContent = "";

    }

    return isValid;

}

* 1. Click on the **Go Live** button to start the live server as shown in the screenshot below:  
       
       
       
     The following page will appear:

A screen shot of a computer

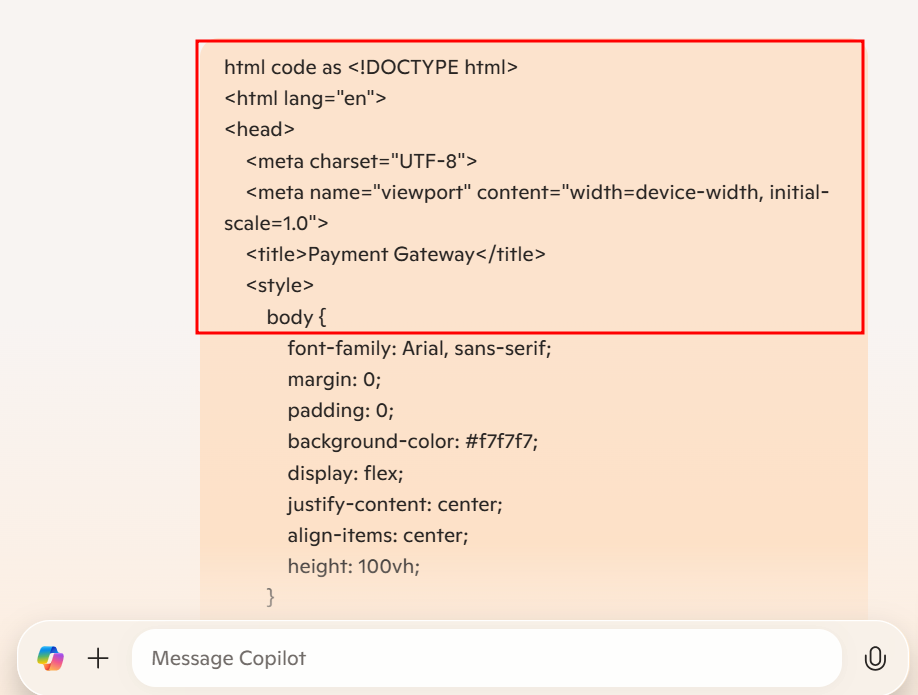
Description automatically generated

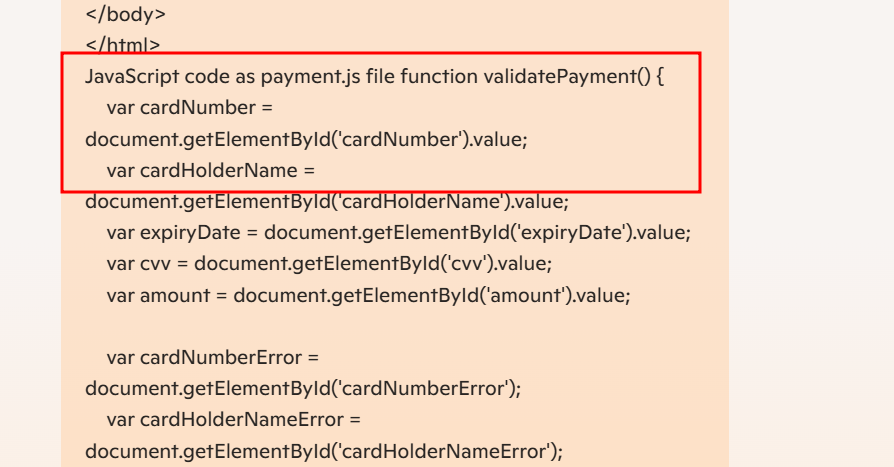
* 1. Open Microsoft Copilot in Google Chrome  
       
     
  2. Add the following prompt in the **Ask me anything…**area and paste the code copied in step 1.4:

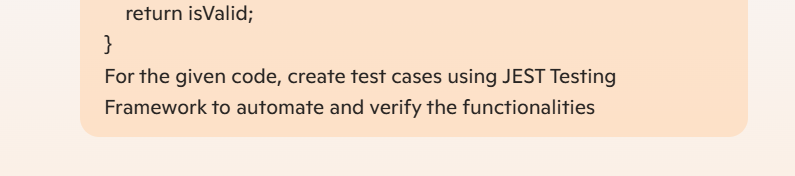
**Html code as (copy and paste complete html code in prompt)**

**JavaScript code as (copy and paste complete html code in prompt)**

**For the given code, create test cases using JEST Testing Framework to automate and verify the functionalities**



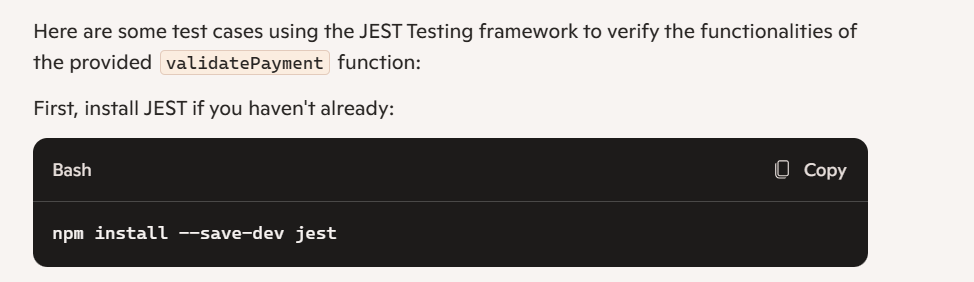




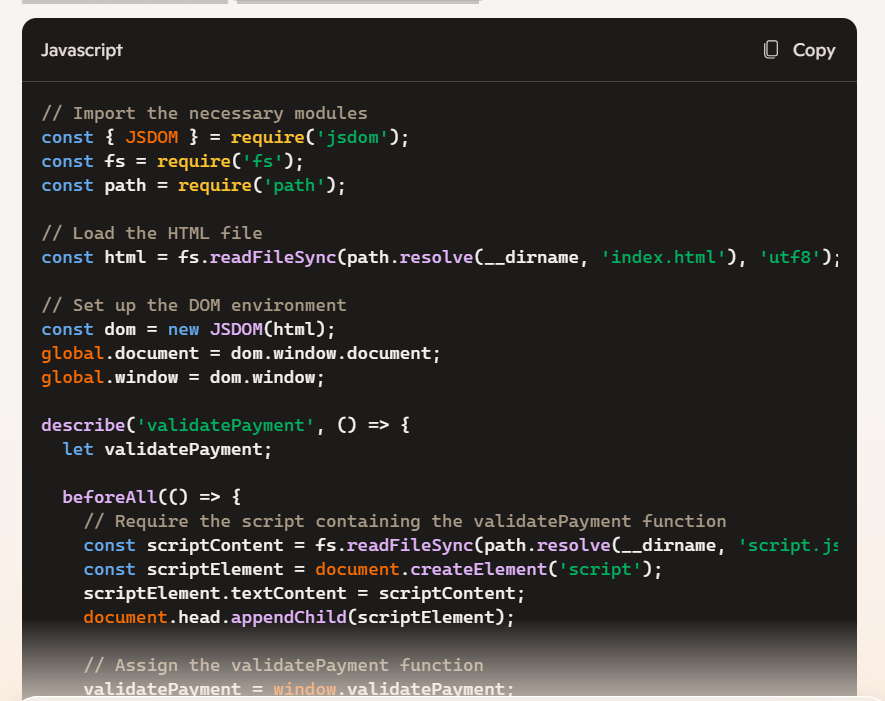
* 1. Click on the **Submit** button to generate the test cases

Here you can see the code generate as

Microsoft Copilot starts generating the test cases as shown below:



Create a test file, e.g., validatePayment.test.js and add the following code:



1.8 First create the package.json file using below command

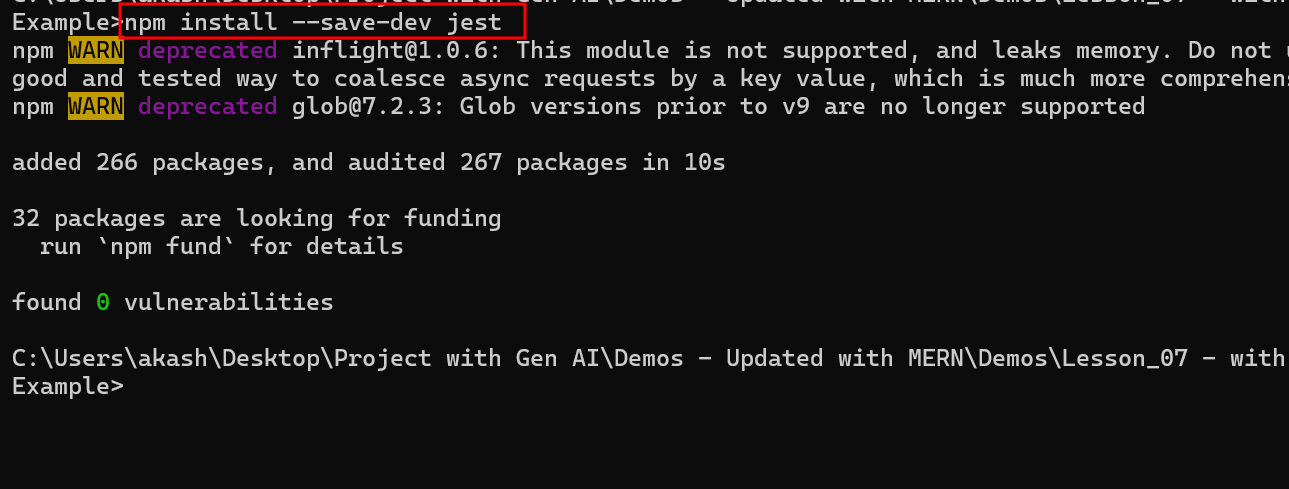
**npm init -y**

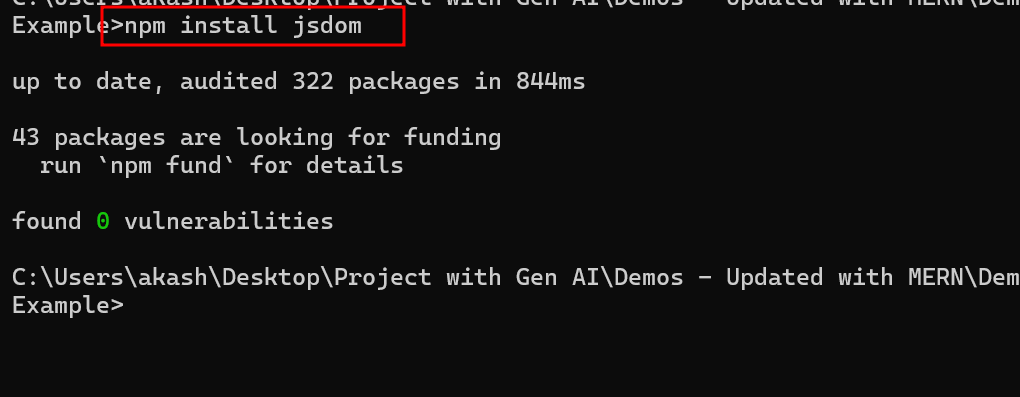


Now install required dependencies

**npm install --save-dev jest**

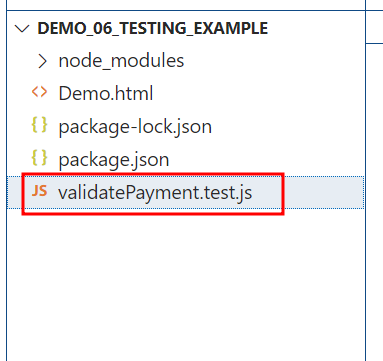
**npm install jsdom**





1.9 Create the file with name as

**validatePayment.test.js**



1.10 Now copy and paste generated code in VS code inside validatePayment.test.js file

validatePayment.test.js file code

const { JSDOM } = require('jsdom');

describe('Payment Gateway Validation', () => {

    let document;

    beforeEach(() => {

        const dom = new JSDOM(`

            <!DOCTYPE html>

            <html lang="en">

            <head>

                <meta charset="UTF-8">

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

                <title>Payment Gateway</title>

                <style>

                    .error-message {

                        display: none;

                    }

                </style>

            </head>

            <body>

                <div class="container">

                    <form id="paymentForm">

                        <input type="text" id="cardNumber" placeholder="Enter your card number">

                        <div class="error-message" id="cardNumberError"></div>

                        <input type="text" id="cardHolderName" placeholder="Enter cardholder name">

                        <div class="error-message" id="cardHolderNameError"></div>

                        <input type="text" id="expiryDate" placeholder="MM/YYYY">

                        <div class="error-message" id="expiryDateError"></div>

                        <input type="text" id="cvv" placeholder="Enter CVV">

                        <div class="error-message" id="cvvError"></div>

                        <input type="number" id="amount" placeholder="Enter amount">

                        <div class="error-message" id="amountError"></div>

                        <button type="submit">Pay Now</button>

                    </form>

                </div>

            </body>

            </html>

        `);

        document = dom.window.document;

        // Mock functions to simulate DOM behavior

        // document.getElementById('cardNumber').value = '';

        // document.getElementById('cardHolderName').value = '';

        // document.getElementById('expiryDate').value = '';

        // document.getElementById('cvv').value = '';

        // document.getElementById('amount').value = '';

        // Provide valid default values for each input

    document.getElementById('cardNumber').value = '1234567812345678';

    document.getElementById('cardHolderName').value = 'John Doe';

    document.getElementById('expiryDate').value = '12/2025';

    document.getElementById('cvv').value = '123';

    document.getElementById('amount').value = '100';

    });

    const validatePayment = () => {

        const cardNumber = document.getElementById('cardNumber').value;

        const cardHolderName = document.getElementById('cardHolderName').value;

        const expiryDate = document.getElementById('expiryDate').value;

        const cvv = document.getElementById('cvv').value;

        const amount = document.getElementById('amount').value;

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

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

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

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

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

        let isValid = true;

        if (!/^(\d{16})$/.test(cardNumber)) {

            cardNumberError.textContent = "Please enter a valid card number.";

            cardNumberError.style.display = 'block';

            isValid = false;

        } else {

            cardNumberError.textContent = "";

            cardNumberError.style.display = 'none';

        }

        if (cardHolderName.trim() === "") {

            cardHolderNameError.textContent = "Please enter the cardholder name.";

            cardHolderNameError.style.display = 'block';

            isValid = false;

        } else {

            cardHolderNameError.textContent = "";

            cardHolderNameError.style.display = 'none';

        }

        if (!/^(0[1-9]|1[0-2])\/\d{4}$/.test(expiryDate)) {

            expiryDateError.textContent = "Please enter a valid expiry date in MM/YYYY format.";

            expiryDateError.style.display = 'block';

            isValid = false;

        } else {

            expiryDateError.textContent = "";

            expiryDateError.style.display = 'none';

        }

        if (!/^\d{3}$/.test(cvv)) {

            cvvError.textContent = "Please enter a valid CVV.";

            cvvError.style.display = 'block';

            isValid = false;

        } else {

            cvvError.textContent = "";

            cvvError.style.display = 'none';

        }

        if (isNaN(amount) || amount <= 0) {

            amountError.textContent = "Please enter a valid amount.";

            amountError.style.display = 'block';

            isValid = false;

        } else {

            amountError.textContent = "";

            amountError.style.display = 'none';

        }

        return isValid;

    };

    test('should validate card number', () => {

        document.getElementById('cardNumber').value = '1234567812345678';

        expect(validatePayment()).toBe(true);

        document.getElementById('cardNumber').value = 'invalid';

        expect(validatePayment()).toBe(false);

        expect(document.getElementById('cardNumberError').textContent).toBe('Please enter a valid card number.');

    });

    test('should validate cardholder name', () => {

        document.getElementById('cardHolderName').value = 'John Doe';

        expect(validatePayment()).toBe(true);

        document.getElementById('cardHolderName').value = '';

        expect(validatePayment()).toBe(false);

        expect(document.getElementById('cardHolderNameError').textContent).toBe('Please enter the cardholder name.');

    });

    test('should validate expiry date', () => {

        document.getElementById('expiryDate').value = '12/2025';

        expect(validatePayment()).toBe(true);

        document.getElementById('expiryDate').value = 'invalid';

        expect(validatePayment()).toBe(false);

        expect(document.getElementById('expiryDateError').textContent).toBe('Please enter a valid expiry date in MM/YYYY format.');

    });

    test('should validate cvv', () => {

        document.getElementById('cvv').value = '123';

        expect(validatePayment()).toBe(true);

        document.getElementById('cvv').value = 'invalid';

        expect(validatePayment()).toBe(false);

        expect(document.getElementById('cvvError').textContent).toBe('Please enter a valid CVV.');

    });

    test('should validate amount', () => {

        document.getElementById('amount').value = '100';

        expect(validatePayment()).toBe(true);

        document.getElementById('amount').value = '-10';

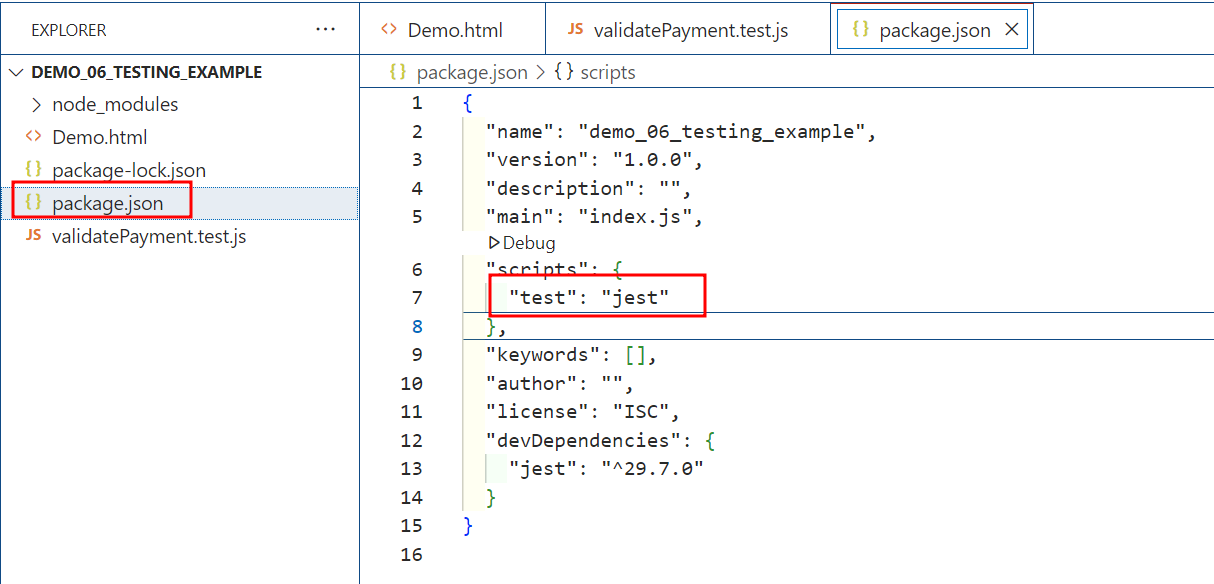
        expect(validatePayment()).toBe(false);

        expect(document.getElementById('amountError').textContent).toBe('Please enter a valid amount.');

    });

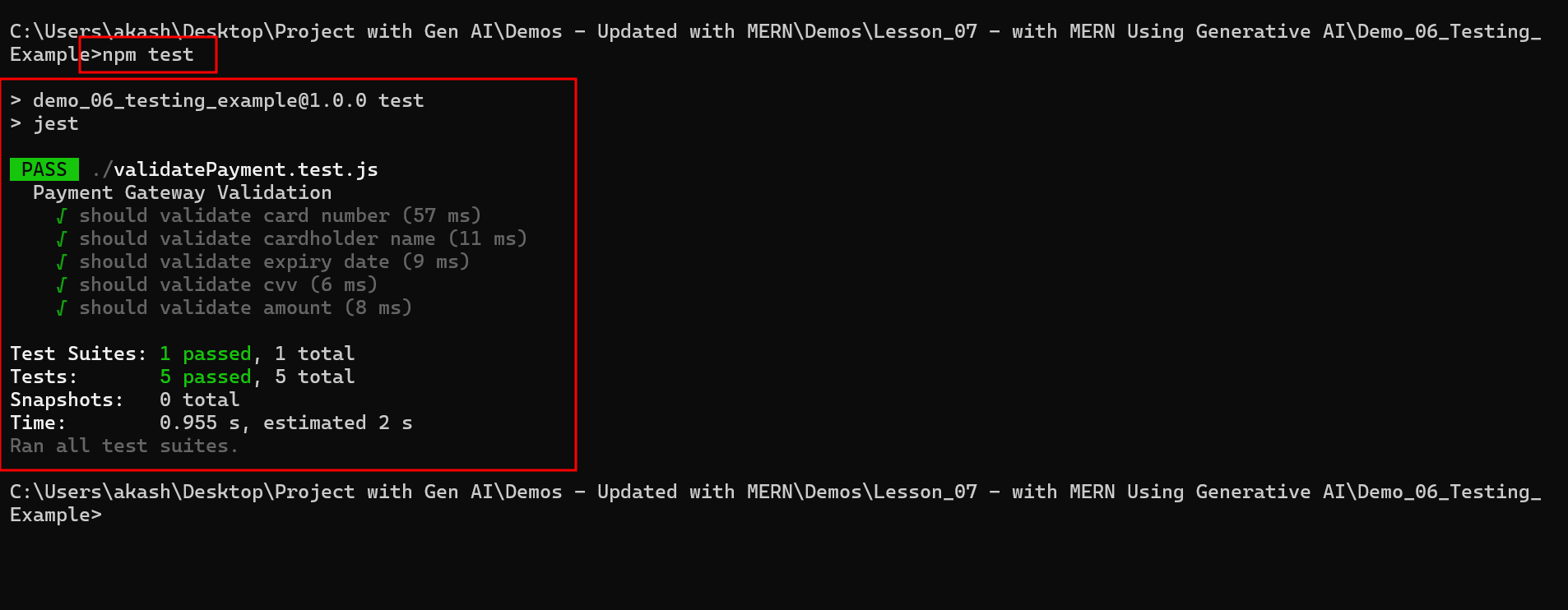
});

1.11 open the package.json file and provide the testing framework details

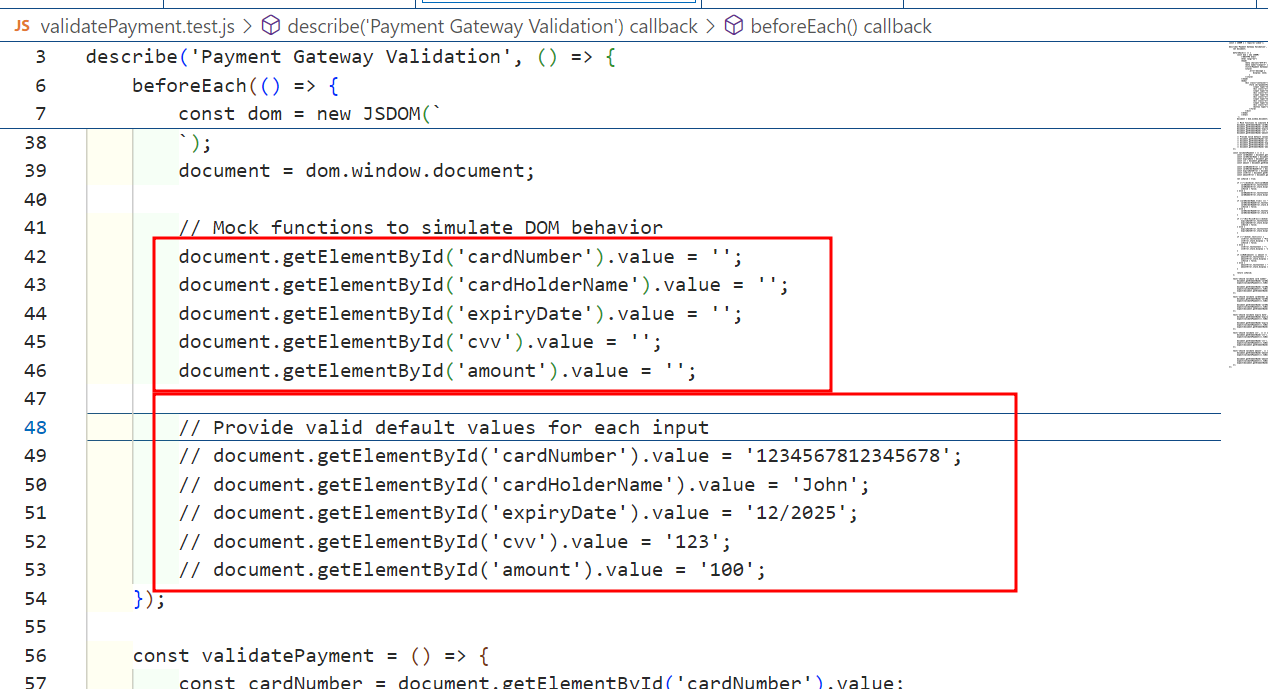


1.12 run the test file using below command as

**npm test**



In code if you set the empty value as



Enable the code 42 to 46 rather than 49 to 53. You will get the validation error.



By following these steps, you have successfully utilized generative AI to create and execute detailed, scenario-based test scripts. This enhances the efficiency and accuracy of validating the functionality and reliability of a payment gateway system and ensures a robust user experience in software development.