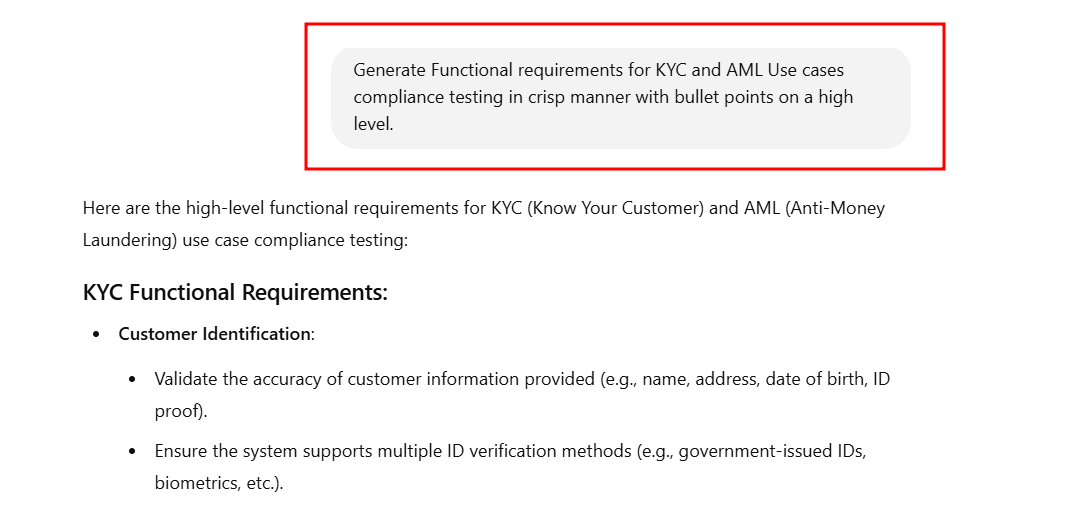
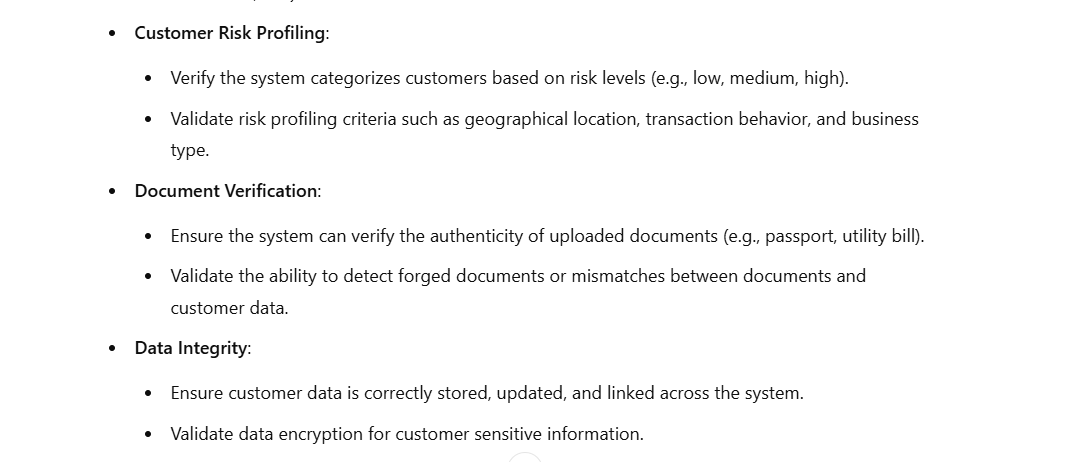
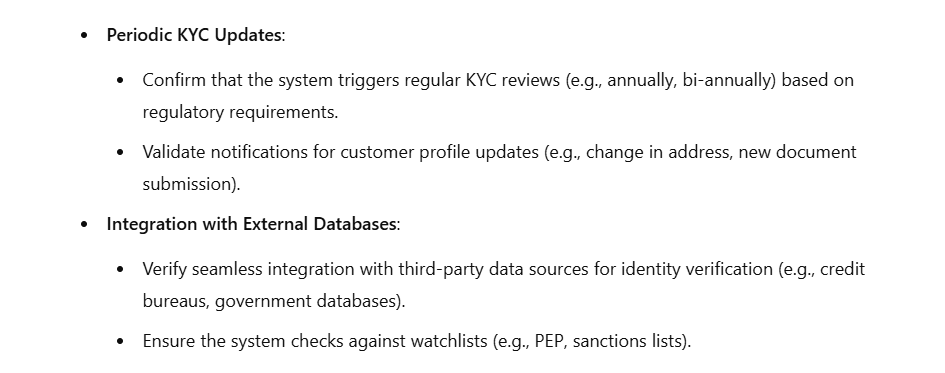
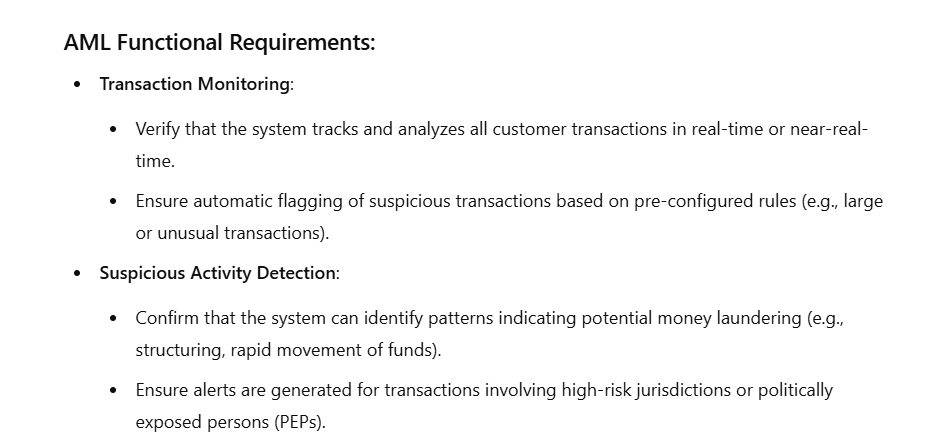
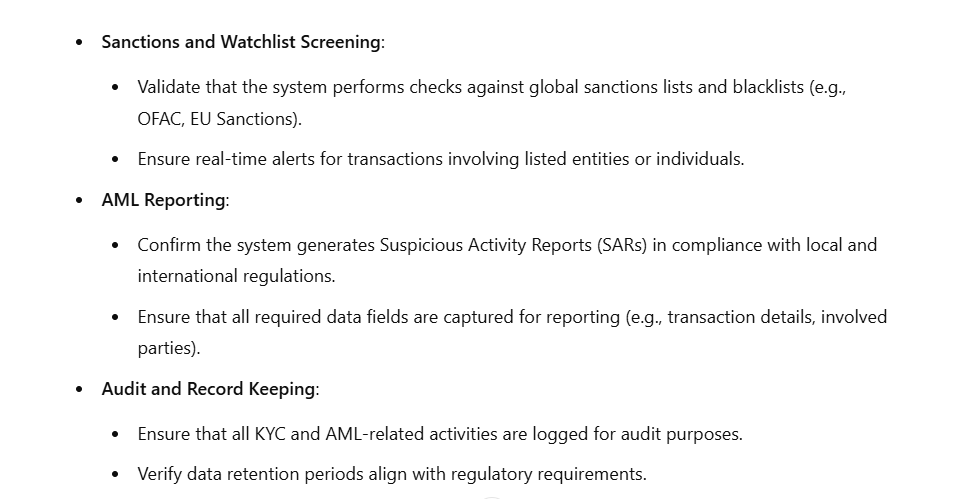
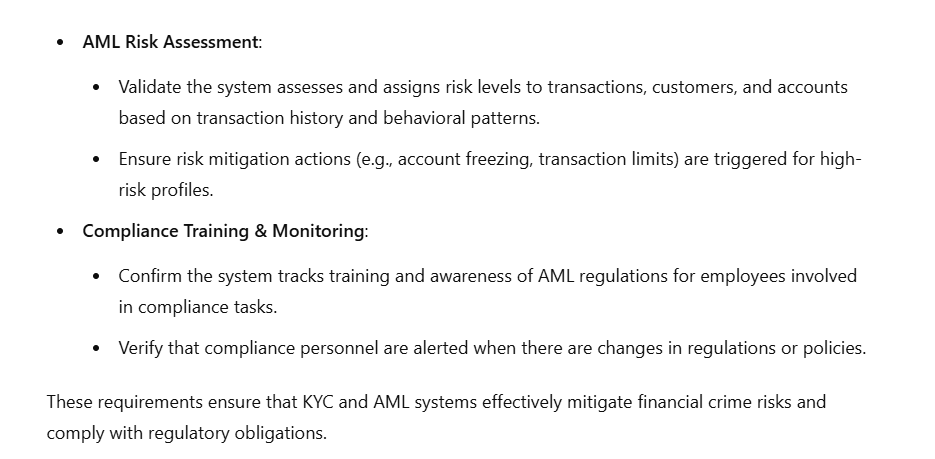
**Answer Key**

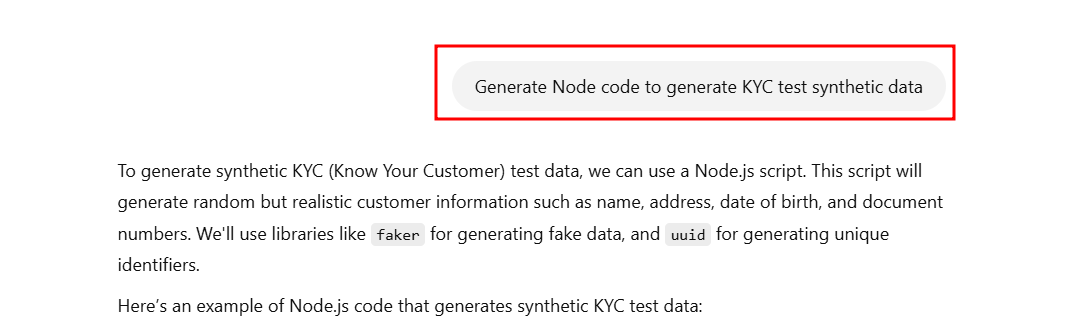
1. **Use prompts to create a comprehensive list of requirements covering chatbot functionality, NLP capabilities, integration with existing systems, scalability needs, and performance expectations**
2. Use the following prompt in ChatGPT to generate functional requirements  
    for KYU and AML

**Generate Functional requirements for KYC and AML Use cases compliance testing in crisp manner with bullet points on a high level**





1. **Generate synthetic data for KYC (Know your customer) use cases**
2. Use the following prompt in ChatGPT to generate Python code for KYC test synthetic data:   
   **Generate Python code to generate KYC test synthetic data**
3. Create the folder and installed required dependencies
4. Install dependencies:

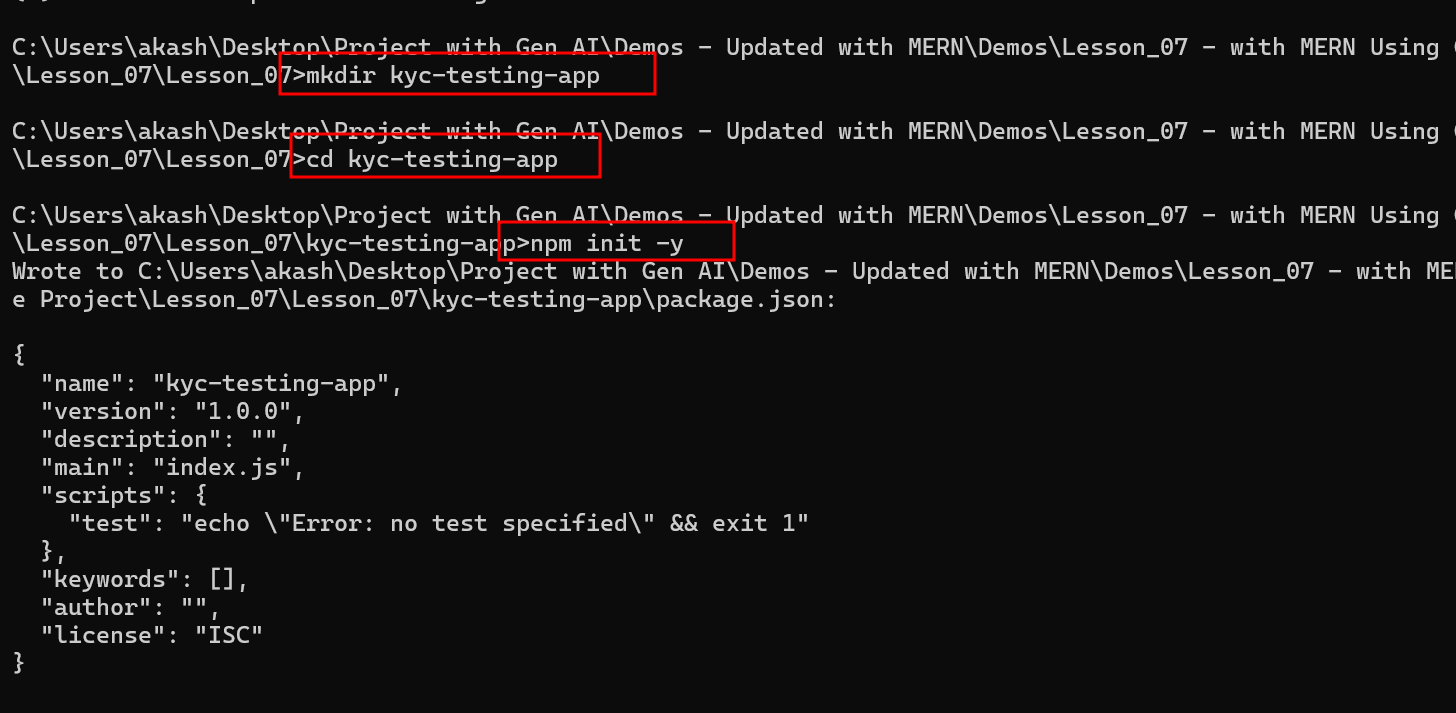
**mkdir kyc-testing-app**

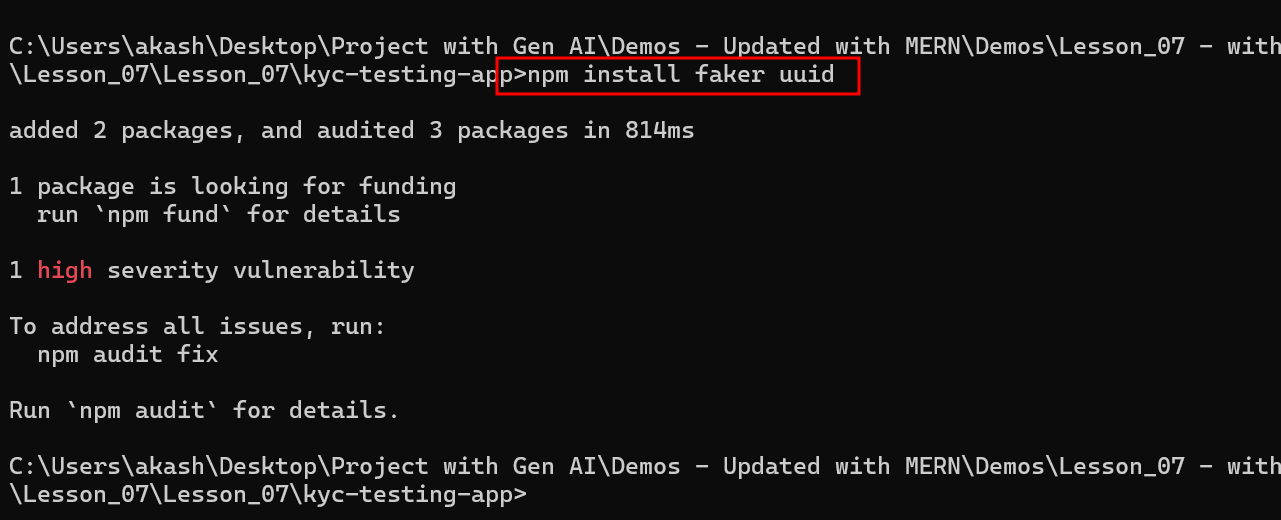
**cd kyc-testing-app**

**npm init -y npm**

**npm install uuid**

**npm install faker@5 --save**





1. Create a file named **generate\_kyc\_data.js** and add the following code:

const faker = require('faker');

const { v4: uuidv4 } = require('uuid');

// Function to generate synthetic KYC data

function generateKycData(numberOfRecords) {

  const kycData = [];

  for (let i = 0; i < numberOfRecords; i++) {

    const data = {

      customerId: uuidv4(), // Generate a unique customer ID

      firstName: faker.name.firstName(), // Random first name

      lastName: faker.name.lastName(), // Random last name

      dateOfBirth: faker.date.past(30, new Date('2000-01-01')), // Random DOB

      address: {

        streetAddress: faker.address.streetAddress(), // Random street address

        city: faker.address.city(), // Random city

        state: faker.address.state(), // Random state

        zipCode: faker.address.zipCode(), // Random zip code

        country: faker.address.country() // Random country

      },

      phoneNumber: faker.phone.phoneNumber(), // Random phone number

      email: faker.internet.email(), // Random email

      nationality: faker.address.countryCode(), // Random nationality code

      document: {

        type: "Passport", // Document type

        number: faker.random.alphaNumeric(9).toUpperCase() // Random passport number

      },

      riskLevel: faker.random.arrayElement(['Low', 'Medium', 'High']), // Random risk level

      createdAt: faker.date.past(2), // Account creation date

      updatedAt: new Date() // Current date

    };

    kycData.push(data);

  }

  return kycData;

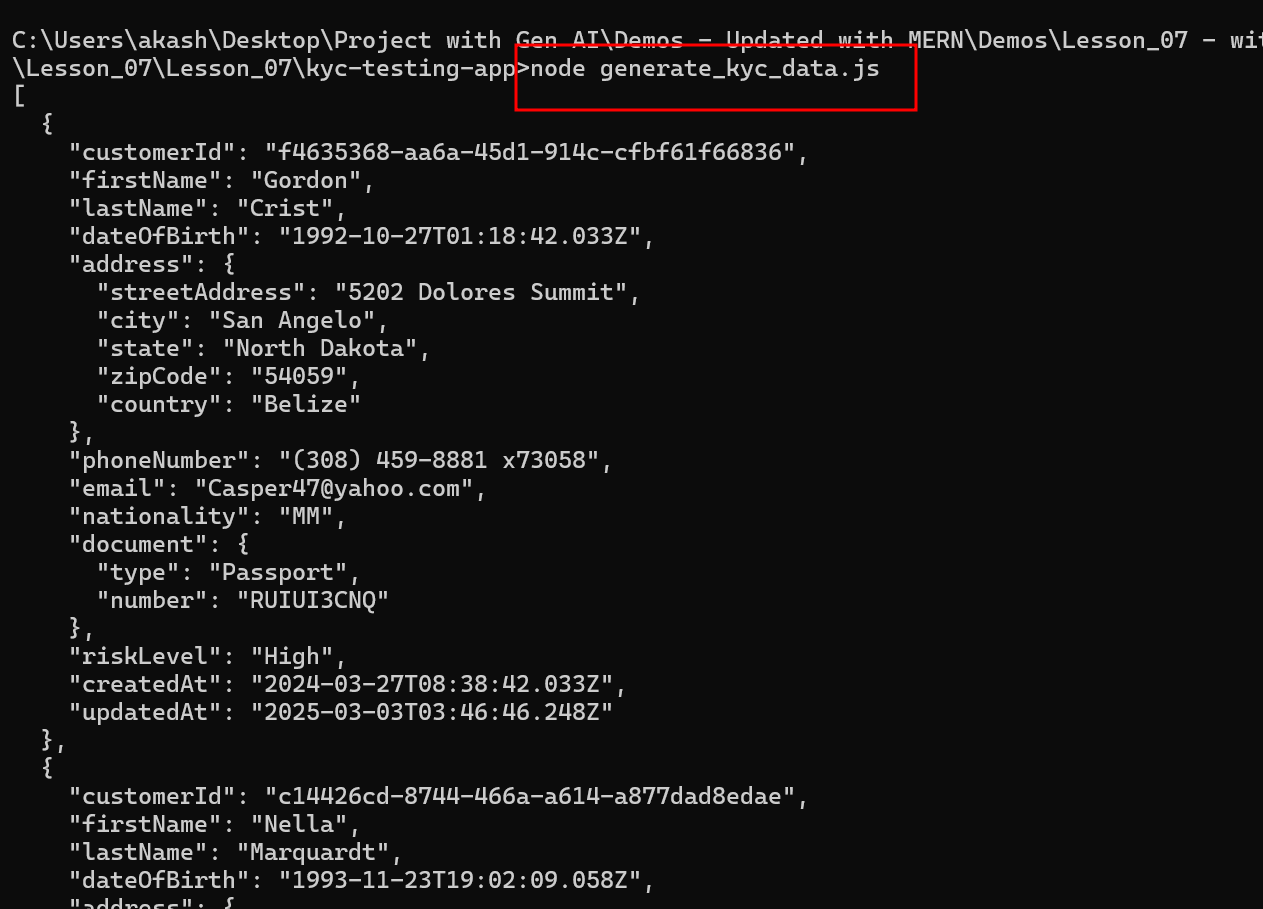
}

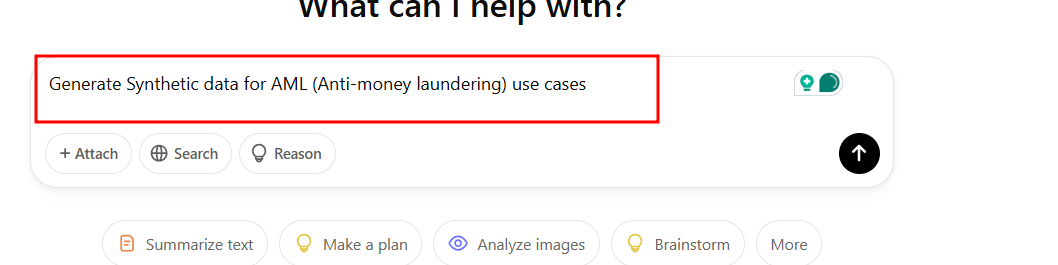
// Generate synthetic data for 10 customers

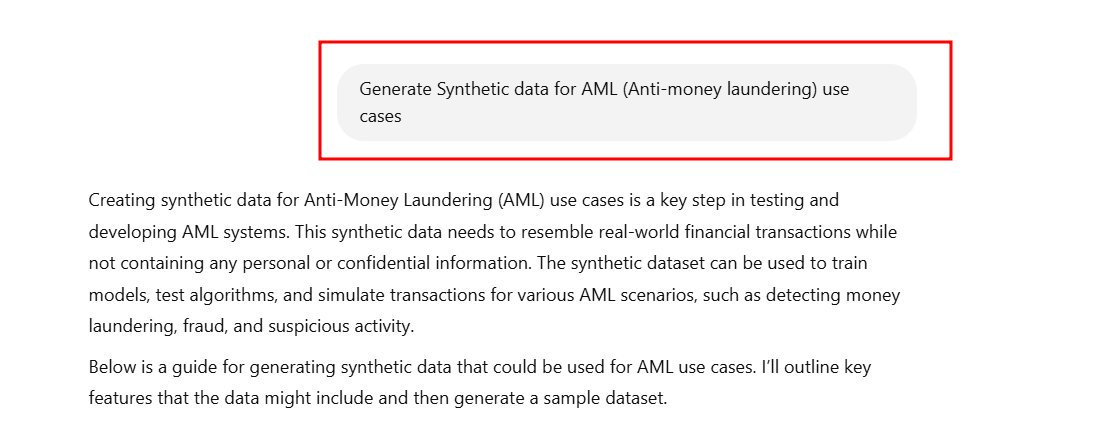
const syntheticKycData = generateKycData(10);

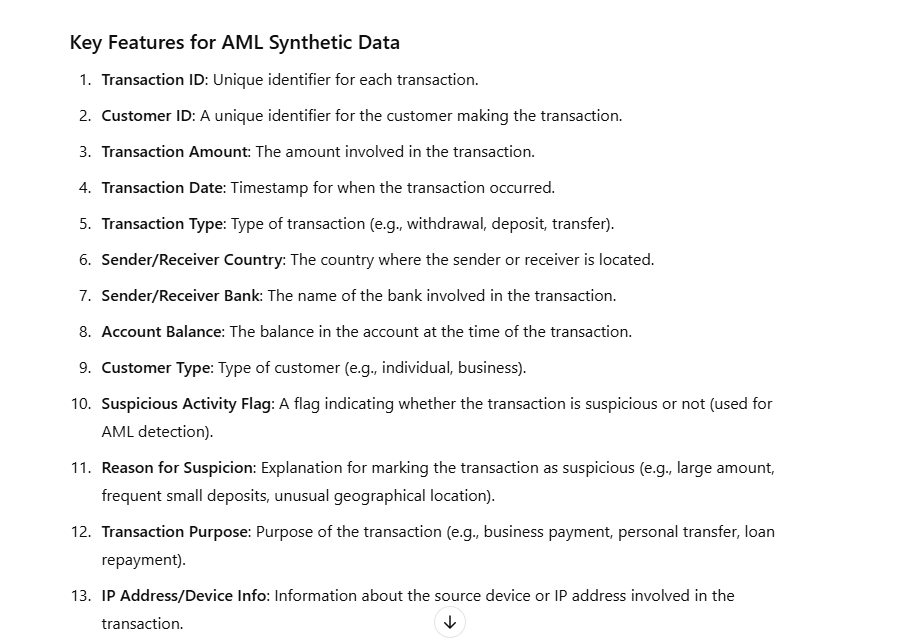
// Output the generated data to the console

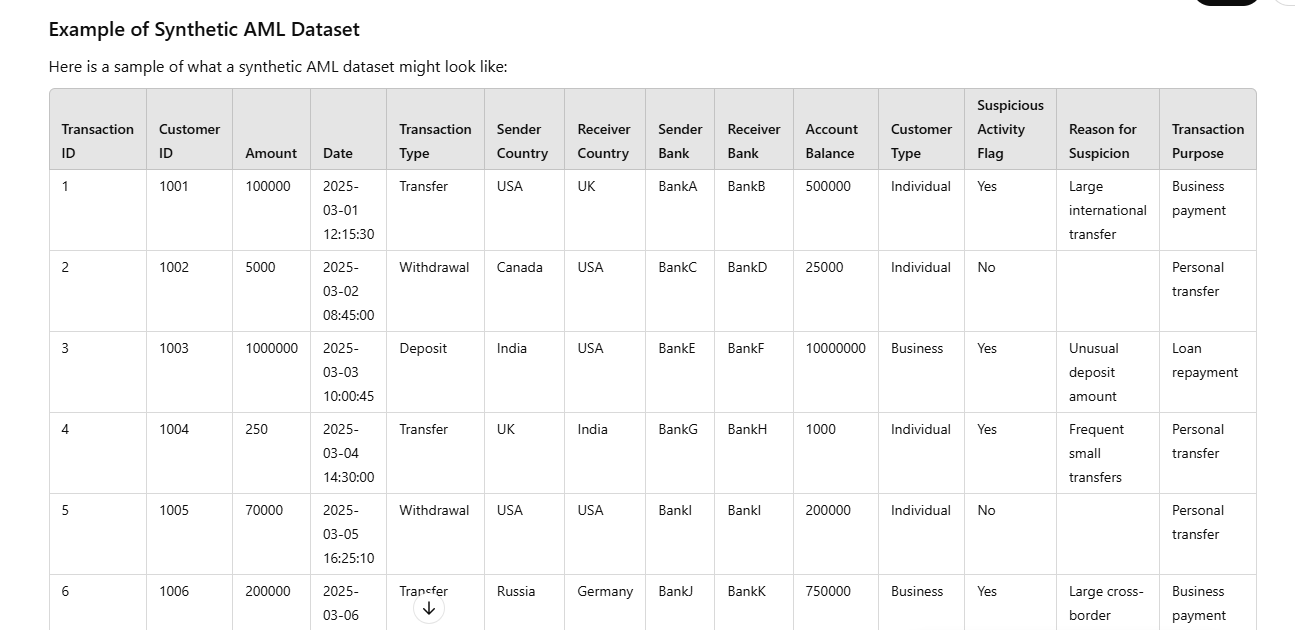
console.log(JSON.stringify(syntheticKycData, null, 2));

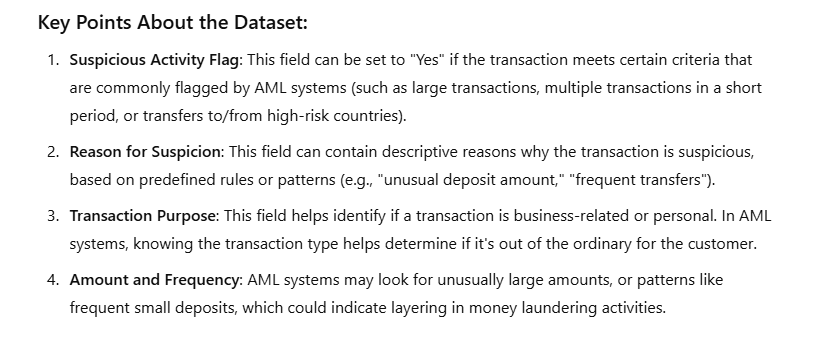


1. **Generate synthetic data for AML (Anti-money laundering) use cases**
2. Use the following prompt to generate synthetic data for AML use cases:   
   **Generate Synthetic data for AML (Anti-money laundering) use cases**  
   



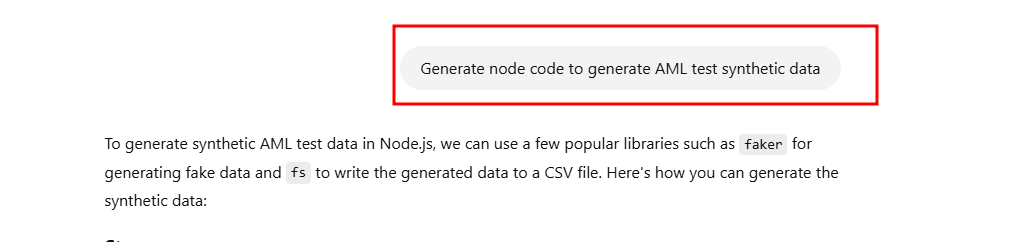






1. Write the following prompt message to generate aml test data.

**Generate node code to generate AML test synthetic data**



1. Create a file named **generate\_aml\_data.js** and add the following code:

const faker = require('faker');

const fs = require('fs');

// Number of records to generate

const numRecords = 1000;

// Transaction types

const transactionTypes = ['Withdrawal', 'Deposit', 'Transfer'];

// Possible countries

const countries = ['USA', 'UK', 'India', 'Canada', 'Germany', 'Russia', 'China', 'Japan', 'Australia'];

// Generate synthetic AML test data

function generateSyntheticData(numRecords) {

    const data = [];

    for (let i = 0; i < numRecords; i++) {

        const transactionId = i + 1;

        const customerId = faker.datatype.number({ min: 1000, max: 9999 }); // Updated to faker.datatype.number

        const amount = faker.finance.amount(100, 1000000, 2);

        const transactionDate = faker.date.past(5);

        const transactionType = transactionTypes[Math.floor(Math.random() \* transactionTypes.length)];

        const senderCountry = countries[Math.floor(Math.random() \* countries.length)];

        const receiverCountry = countries[Math.floor(Math.random() \* countries.length)];

        const senderBank = faker.company.companyName();

        const receiverBank = faker.company.companyName();

        const accountBalance = faker.finance.amount(5000, 10000000, 2);

        const customerType = Math.random() > 0.5 ? 'Individual' : 'Business';

        // Suspicious Activity Logic (simple rule-based flag for large amounts)

        let suspiciousActivityFlag = 'No';

        let reasonForSuspicion = '';

        if (parseFloat(amount) > 50000) {

            suspiciousActivityFlag = 'Yes';

            reasonForSuspicion = 'Large transaction amount';

        } else if (transactionType === 'Transfer' && senderCountry !== receiverCountry) {

            suspiciousActivityFlag = 'Yes';

            reasonForSuspicion = 'Large cross-border transfer';

        }

        const transactionPurpose = ['Business payment', 'Personal transfer', 'Loan repayment'][Math.floor(Math.random() \* 3)];

        // Append record to the data array

        data.push([

            transactionId,

            customerId,

            amount,

            transactionDate.toISOString(),

            transactionType,

            senderCountry,

            receiverCountry,

            senderBank,

            receiverBank,

            accountBalance,

            customerType,

            suspiciousActivityFlag,

            reasonForSuspicion,

            transactionPurpose

        ]);

    }

    return data;

}

// Generate the data

const amlData = generateSyntheticData(numRecords);

// Create a CSV file to store the data

const csvHeader = [

    'Transaction ID', 'Customer ID', 'Amount', 'Date', 'Transaction Type',

    'Sender Country', 'Receiver Country', 'Sender Bank', 'Receiver Bank',

    'Account Balance', 'Customer Type', 'Suspicious Activity Flag', 'Reason for Suspicion', 'Transaction Purpose'

].join(',') + '\n';

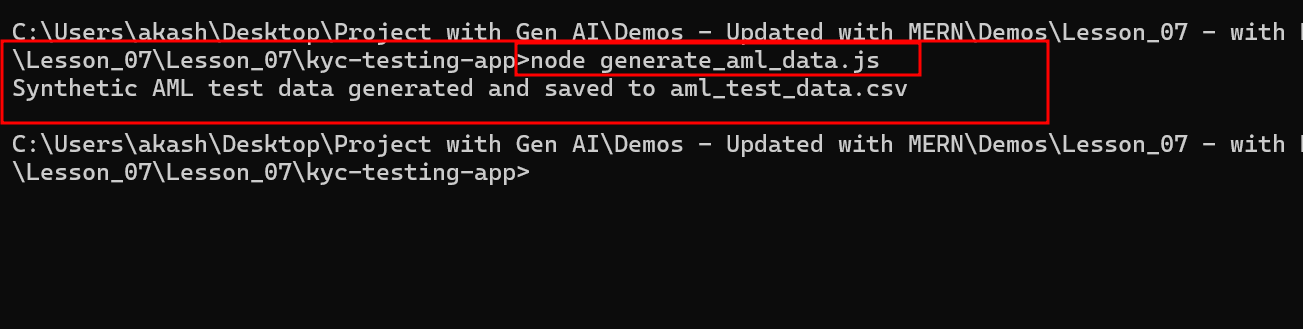
// Convert data to CSV format

const csvData = amlData.map(row => row.join(',')).join('\n');

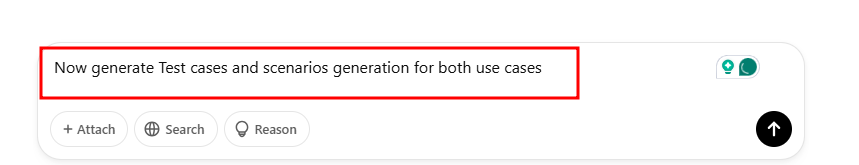
// Write the CSV data to a file

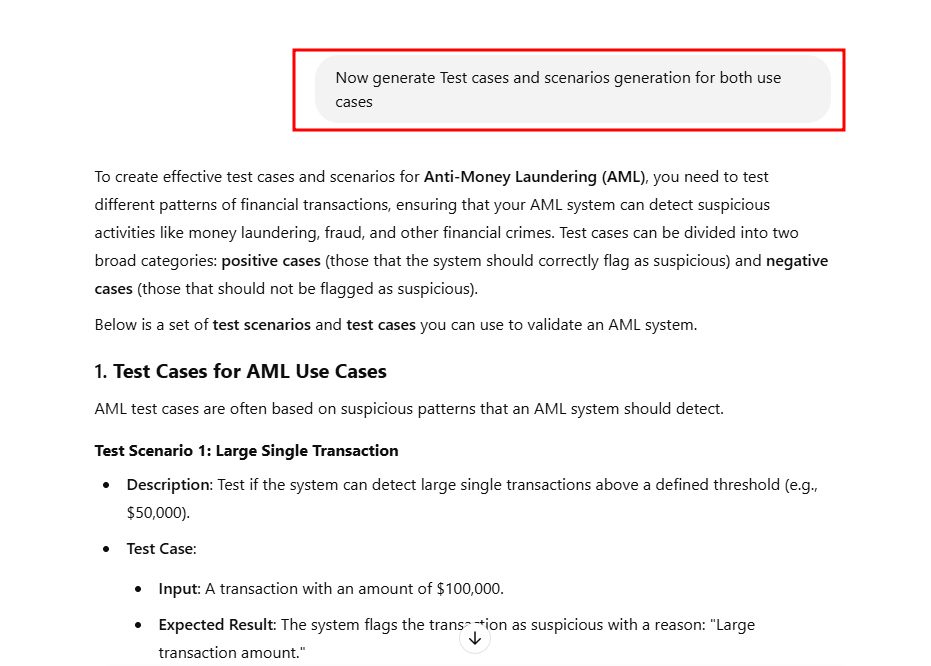
fs.writeFileSync('aml\_test\_data.csv', csvHeader + csvData);

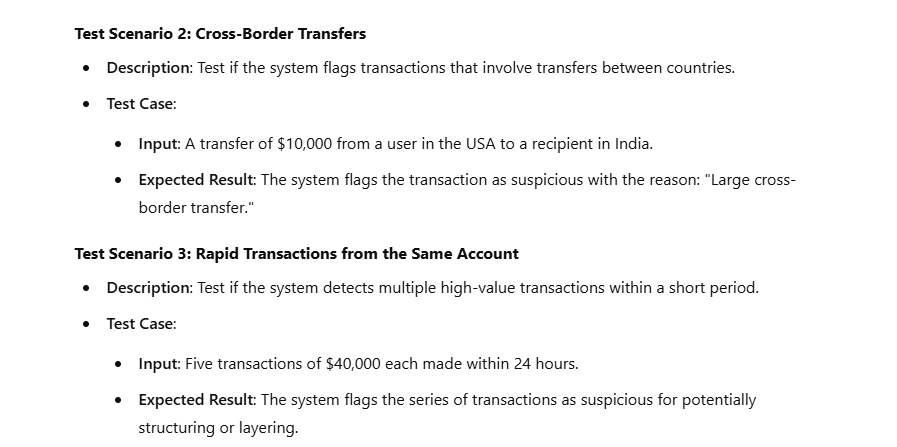
console.log('Synthetic AML test data generated and saved to aml\_test\_data.csv');

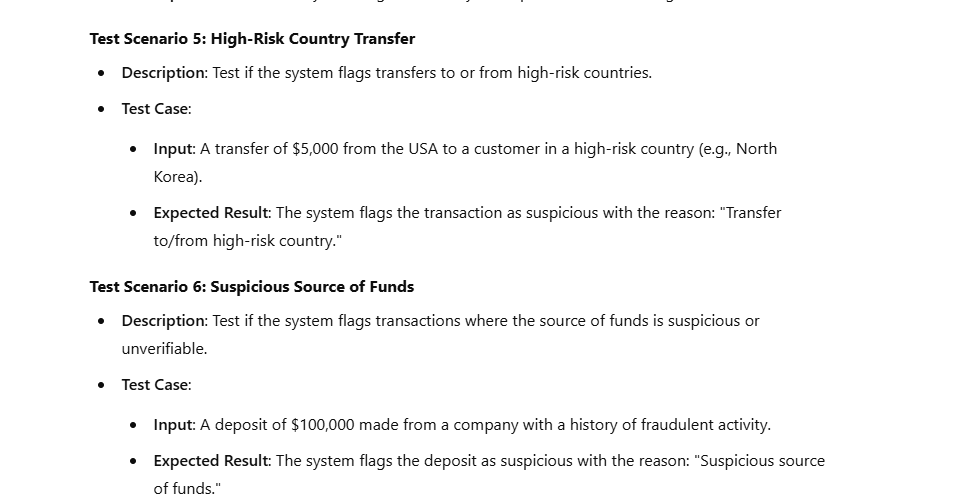
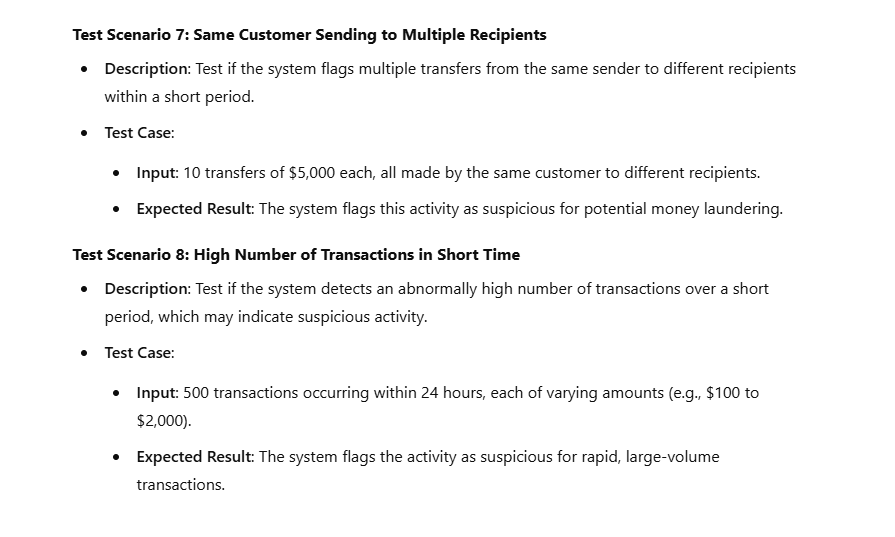


1. **Utilize prompts to generate a test cases and scenarios generation**
2. Use the following prompt to generate the test cases:  
   **now generate Test cases and scenarios generation for Both use cases**







  **2. Automated Test Case Generation with Scenarios**

Here is an example of how you might automate test case generation based on the synthetic data we created in the previous section.

**automated\_test\_case.js**

const faker = require('faker');

// Sample Suspicious Transaction Logic

const testScenarios = [

{

name: "Large Single Transaction",

description: "Test for large single transactions",

scenario: (transaction) => transaction.amount > 50000,

reason: "Large transaction amount"

},

{

name: "Cross-Border Transfer",

description: "Test for transfers between different countries",

scenario: (transaction) => transaction.senderCountry !== transaction.receiverCountry,

reason: "Large cross-border transfer"

},

{

name: "Rapid Transactions",

description: "Test for rapid transactions from the same account",

scenario: (transaction, allTransactions) => {

const recentTransactions = allTransactions.filter(t => t.customerId === transaction.customerId && (Math.abs(new Date(t.date) - new Date(transaction.date)) <= 24 \* 60 \* 60 \* 1000));

return recentTransactions.length > 3 && recentTransactions.reduce((acc, t) => acc + parseFloat(t.amount), 0) > 100000;

},

reason: "Multiple large transactions in short time"

},

{

name: "Frequent Small Transactions",

description: "Test for frequent small transactions from one customer",

scenario: (transaction, allTransactions) => {

const recentTransactions = allTransactions.filter(t => t.customerId === transaction.customerId && (Math.abs(new Date(t.date) - new Date(transaction.date)) <= 24 \* 60 \* 60 \* 1000));

return recentTransactions.length > 10 && recentTransactions.every(t => parseFloat(t.amount) < 200);

},

reason: "Frequent small transactions"

},

{

name: "Transfer to High-Risk Country",

description: "Test for transfer to high-risk countries",

scenario: (transaction) => ['North Korea', 'Iran'].includes(transaction.receiverCountry),

reason: "Transfer to/from high-risk country"

}

];

// Simulate synthetic transactions (sample data)

const transactions = [

{ amount: "60000", customerId: "1234", date: "2025-03-01T10:00:00", senderCountry: "USA", receiverCountry: "UK", senderBank: "BankA", receiverBank: "BankB" },

{ amount: "30000", customerId: "1234", date: "2025-03-01T11:00:00", senderCountry: "USA", receiverCountry: "Canada", senderBank: "BankA", receiverBank: "BankC" },

{ amount: "1000", customerId: "5678", date: "2025-03-02T12:00:00", senderCountry: "Canada", receiverCountry: "USA", senderBank: "BankD", receiverBank: "BankE" }

];

// Function to apply test scenarios to each transaction

function applyTestScenarios(transactions) {

transactions.forEach(transaction => {

testScenarios.forEach(scenario => {

const isSuspicious = scenario.scenario(transaction, transactions);

if (isSuspicious) {

console.log(`Transaction ID: ${transaction.customerId} is suspicious for ${scenario.name}`);

console.log(`Reason: ${scenario.reason}`);

}

});

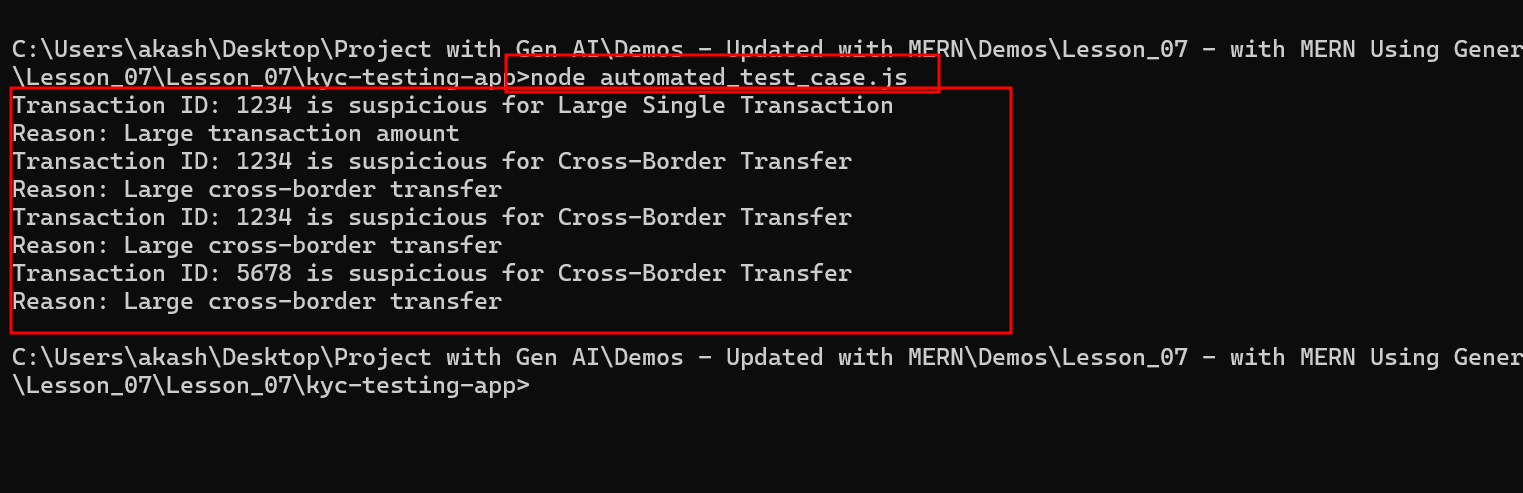
});

}

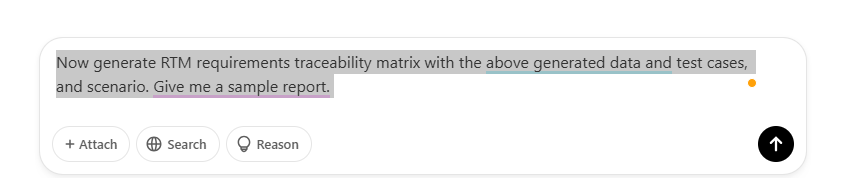
// Apply the test scenarios

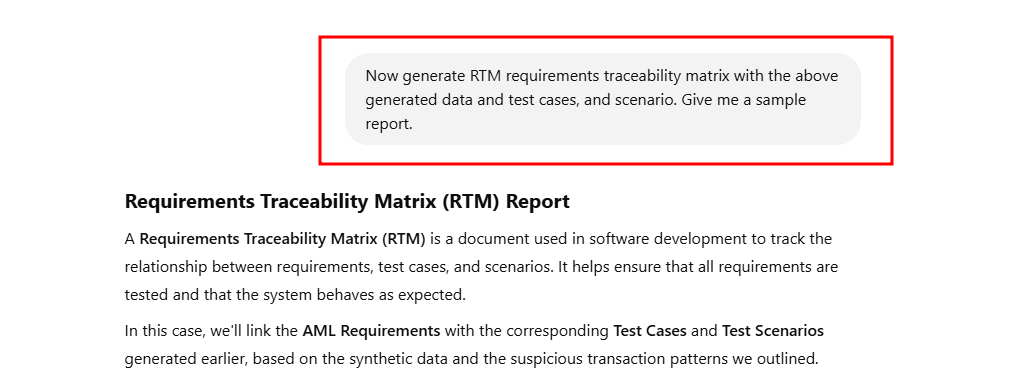
applyTestScenarios(transactions);

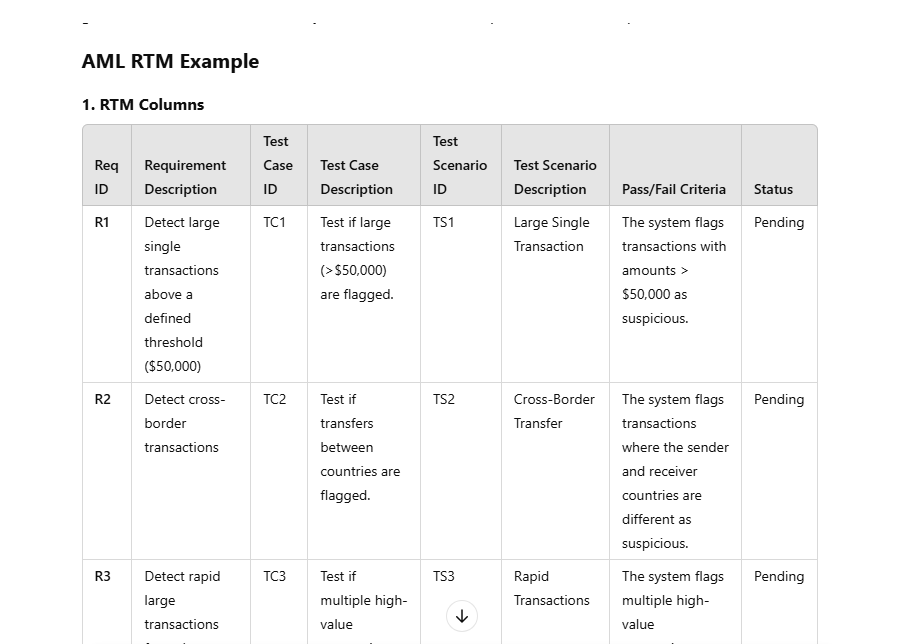
run the application as

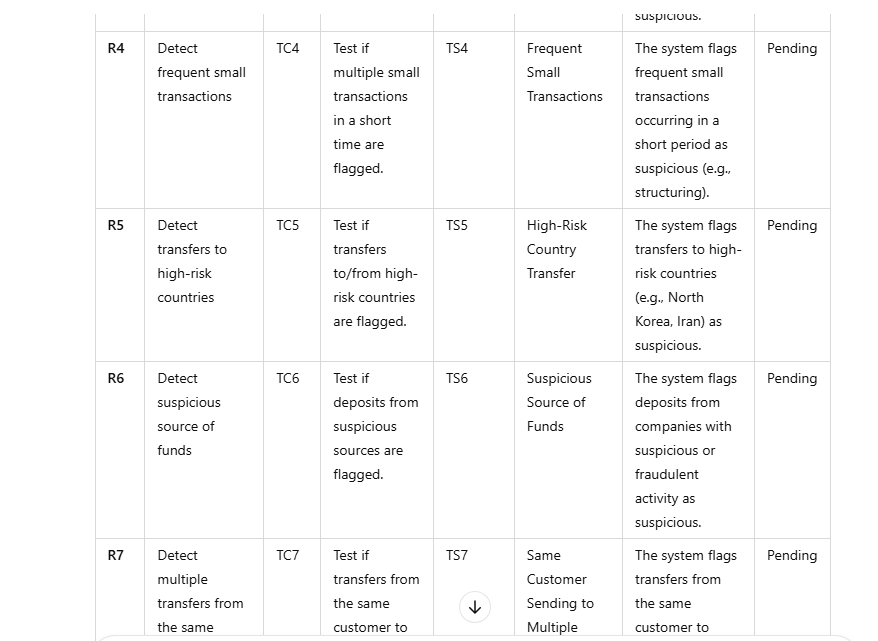
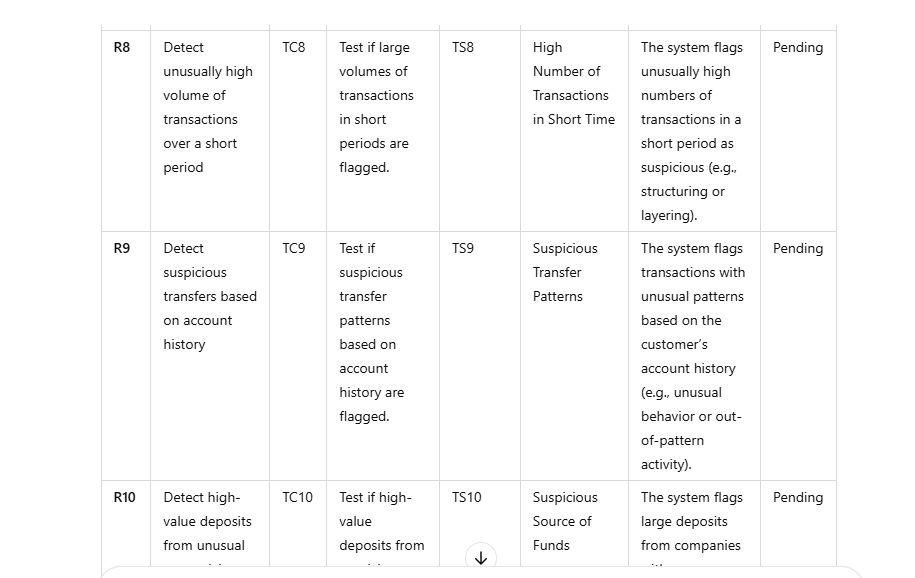
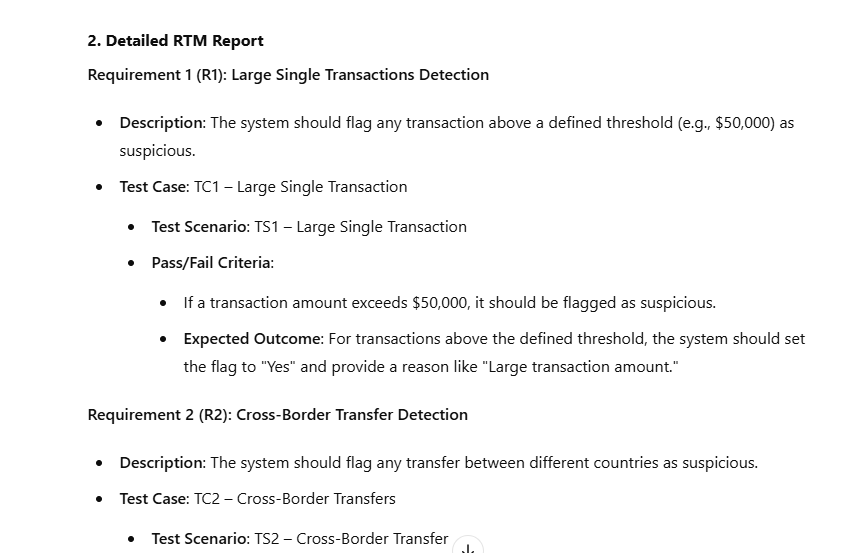
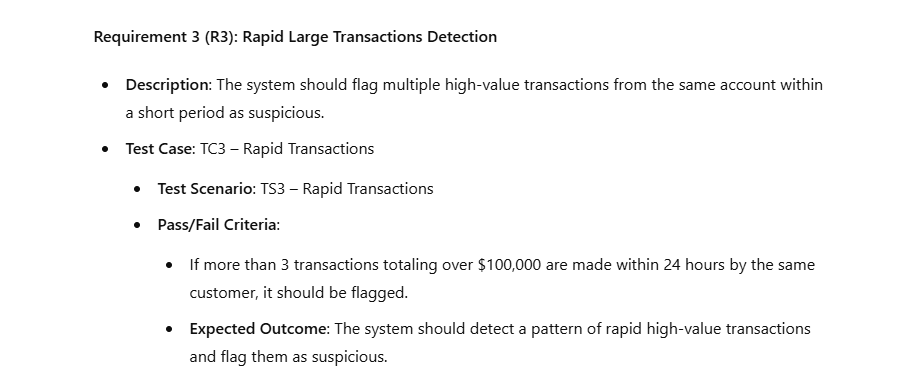
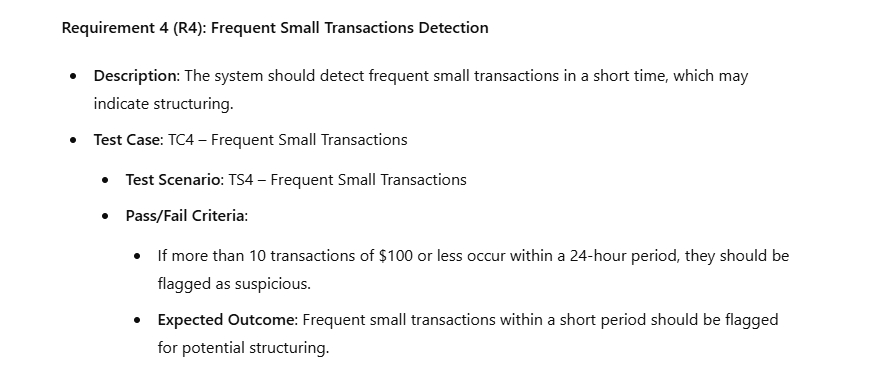
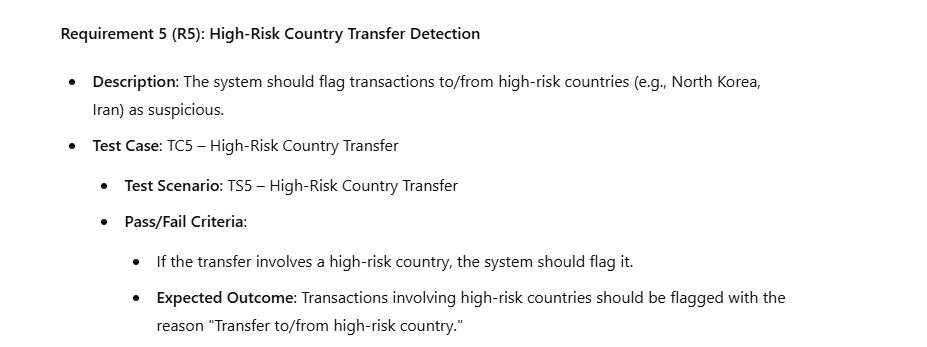
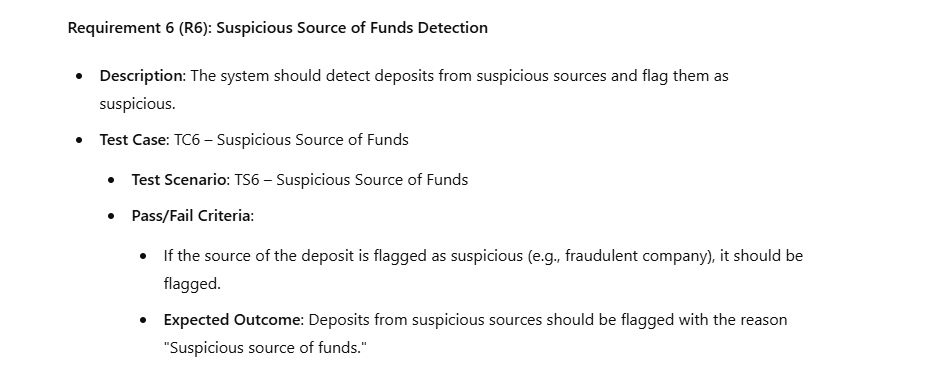
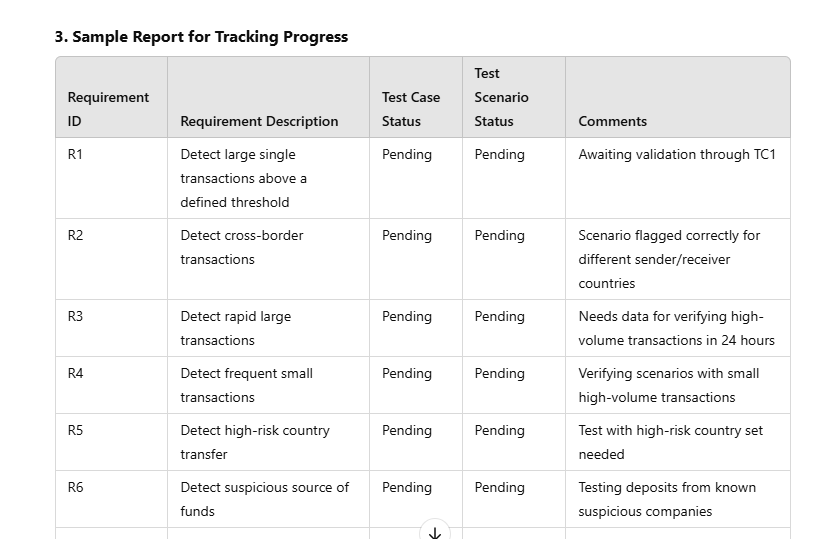
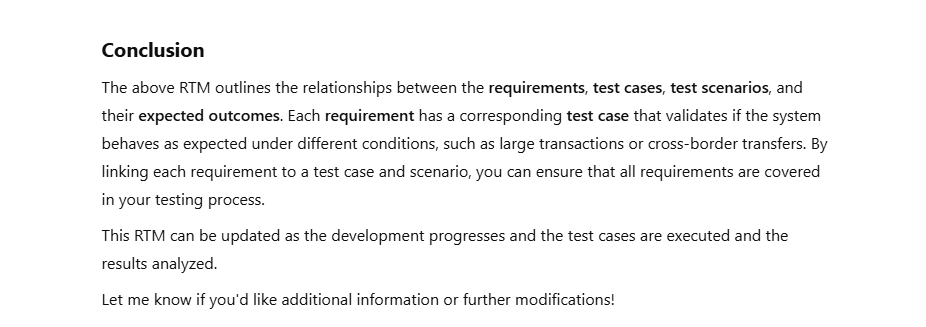


1. **Develop detailed test report mapping with RTM on the above test cases**
2. Use the following prompt to generate test report mapping with RTM:  
   **Now generate RTM requirements traceability matrix with the above generated data and test cases, and scenario. Give me a sample report.**

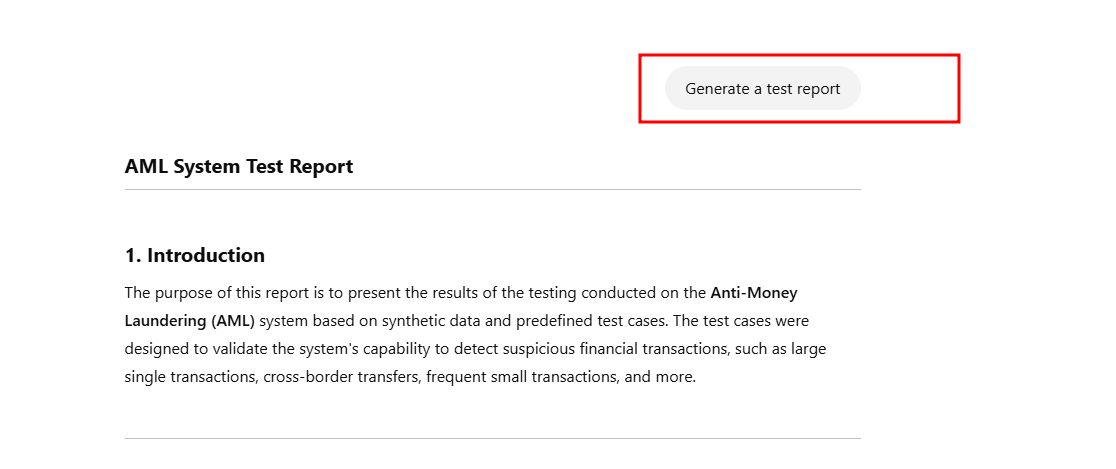
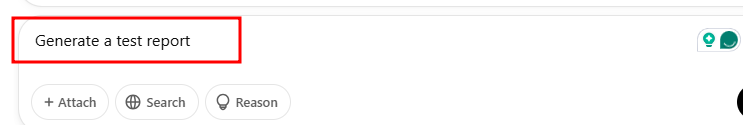


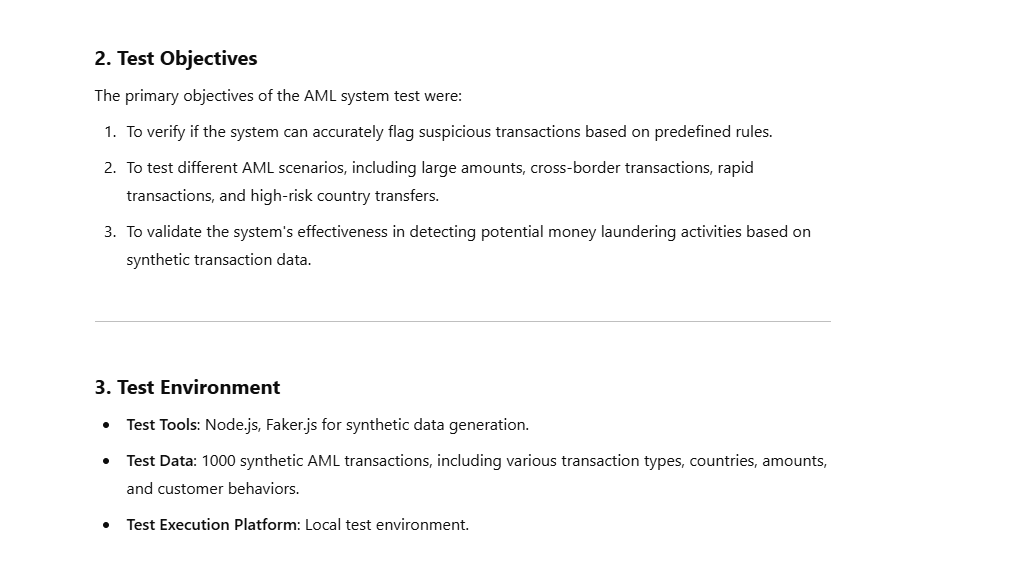


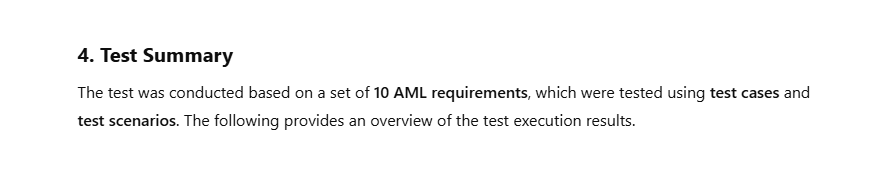
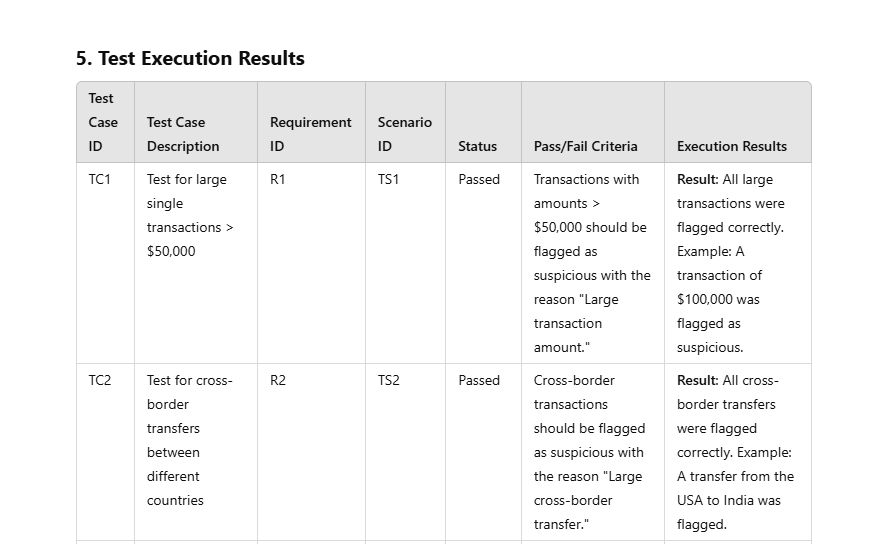
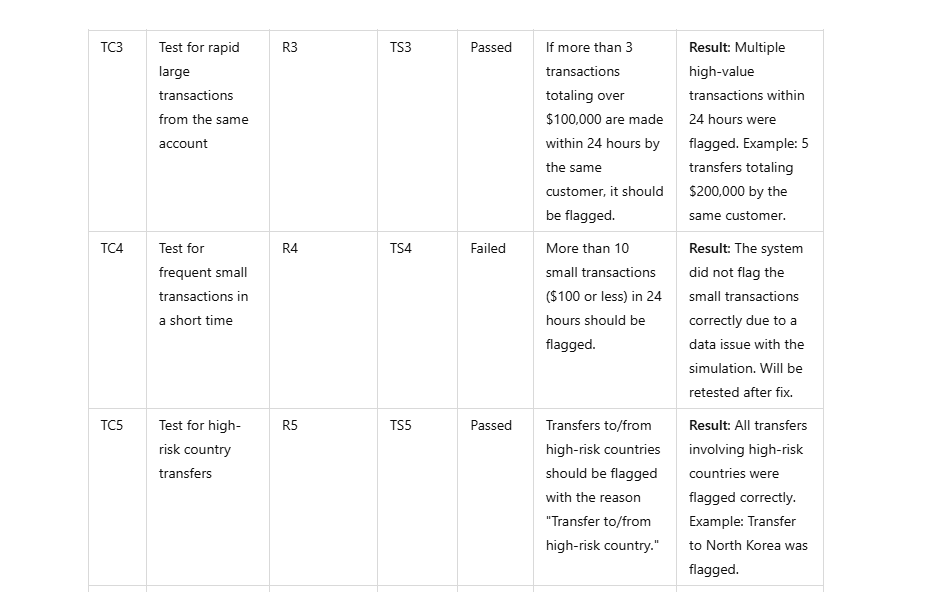
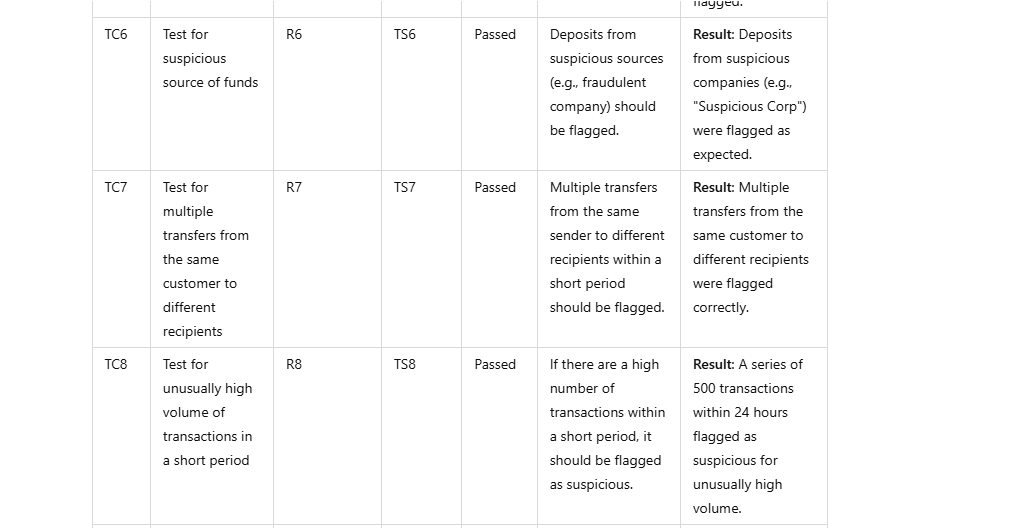


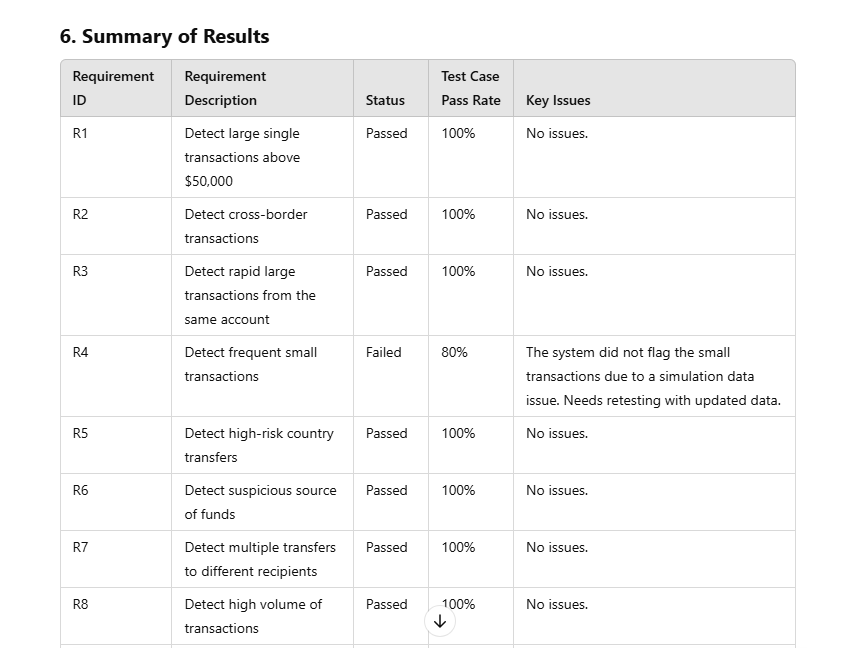
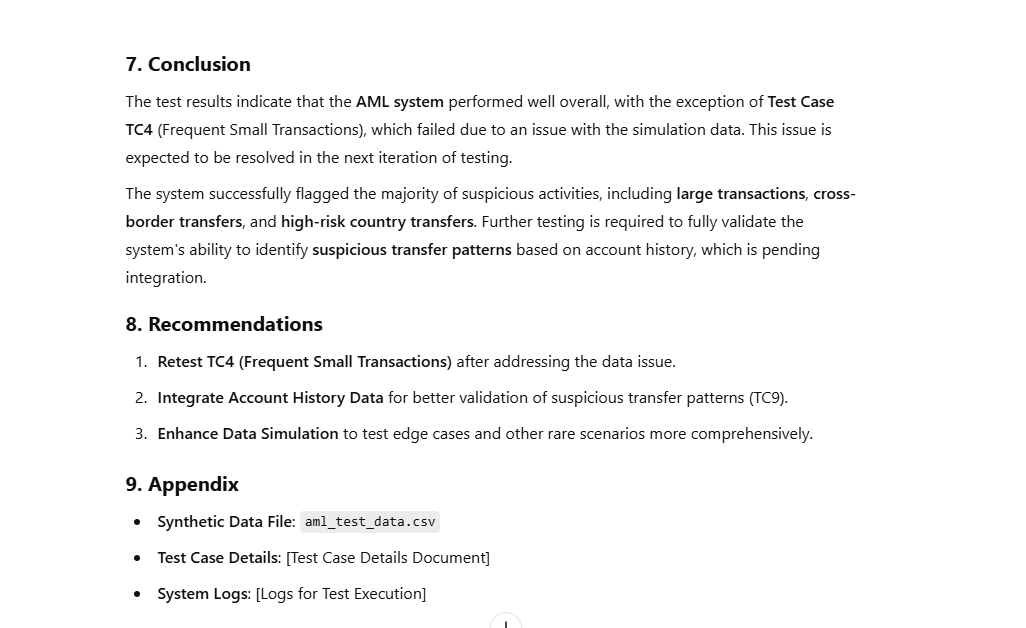
1. Use to following prompt to generate test report:  
   **Generate a test report**



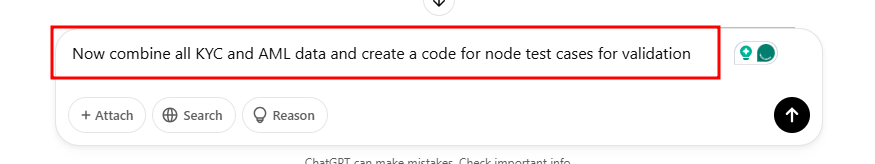


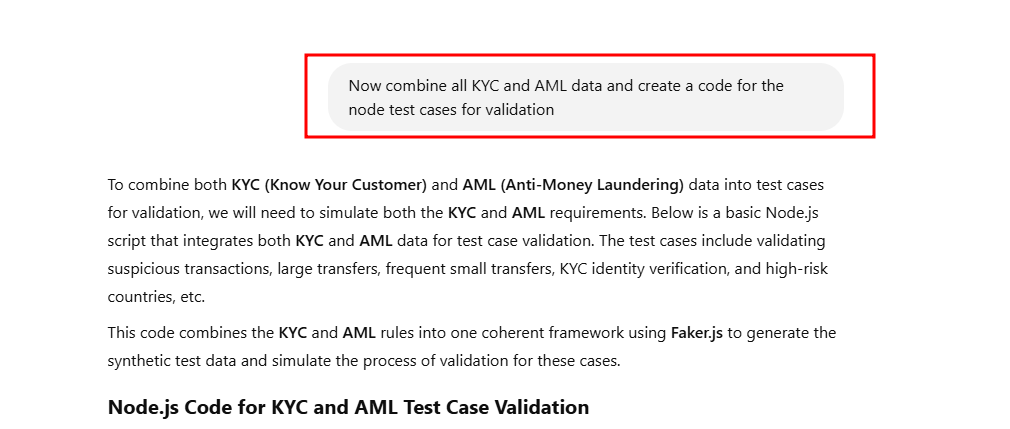
   



1. Use the following command to combine KYC and AML:  
   **Now combine all KYC and AML data and create a code for the node test cases for validation**





Create the file as **kyc\_aml\_test\_validation.js**

1. Create a file named **kyc\_aml\_test\_validation.js** and add the following code:

const faker = require('faker');

// Sample list of high-risk countries (For AML)

const highRiskCountries = ['North Korea', 'Iran', 'Sudan', 'Syria', 'Afghanistan'];

// KYC Data Generation Function (For KYC)

function generateKYCData() {

  return {

    customerId: faker.datatype.uuid(),

    firstName: faker.name.firstName(),

    lastName: faker.name.lastName(),

    dateOfBirth: faker.date.past(30, '2000-01-01'),

    nationality: faker.address.country(),

    address: faker.address.streetAddress(),

    email: faker.internet.email(),

    phone: faker.phone.phoneNumber(),

    identificationNumber: faker.datatype.number({ min: 1000000, max: 9999999 }).toString(),

    occupation: faker.name.jobTitle(),

    riskLevel: faker.random.arrayElement(['Low', 'Medium', 'High']),

  };

}

// AML Data Generation Function (For AML)

function generateAMLTransaction() {

  return {

    transactionId: faker.datatype.uuid(),

    senderId: faker.datatype.uuid(),

    receiverId: faker.datatype.uuid(),

    senderCountry: faker.address.country(),

    receiverCountry: faker.address.country(),

    amount: faker.finance.amount(100, 1000000, 2), // Amount in USD

    transactionDate: faker.date.past(1),

    transactionType: faker.random.arrayElement(['Deposit', 'Withdrawal', 'Transfer']),

    senderAccountHistory: faker.datatype.number({ min: 1, max: 10000 }),

    transactionStatus: 'Completed', // Can be 'Pending' or 'Completed'

  };

}

// Function to validate KYC Data

function validateKYC(kycData) {

  let validationResult = [];

  if (!kycData.firstName || !kycData.lastName || !kycData.dateOfBirth) {

    validationResult.push('Missing required customer identity information.');

  }

  if (!highRiskCountries.includes(kycData.nationality)) {

    validationResult.push('Customer nationality is not flagged as a high-risk country.');

  }

  if (kycData.riskLevel === 'High') {

    validationResult.push('Customer risk level is HIGH, further validation required.');

  }

  return validationResult.length ? validationResult : ['KYC data is valid.'];

}

// Function to validate AML Transactions

function validateAMLTransaction(transactionData, customerKYCData) {

  let validationResult = [];

  // 1. Check for large transactions (greater than $50,000)

  if (parseFloat(transactionData.amount) > 50000) {

    validationResult.push('Transaction exceeds $50,000, flagged for review.');

  }

  // 2. Check for cross-border transactions

  if (transactionData.senderCountry !== transactionData.receiverCountry) {

    validationResult.push('Cross-border transaction detected, flagged for review.');

  }

  // 3. Check for suspicious rapid transactions

  if (transactionData.senderAccountHistory > 1000) {

    validationResult.push('Multiple high-volume transactions detected in a short time.');

  }

  // 4. Check if the sender is from a high-risk country

  if (highRiskCountries.includes(transactionData.senderCountry)) {

    validationResult.push('Sender is from a high-risk country, flagged for review.');

  }

  // 5. Validate KYC data for suspicious activity

  const kycValidation = validateKYC(customerKYCData);

  if (kycValidation[0] !== 'KYC data is valid.') {

    validationResult.push('KYC validation failed. Issues found in customer data.');

    validationResult.push(...kycValidation);

  }

  return validationResult.length ? validationResult : ['Transaction passed AML validation.'];

}

// Sample Data Generation and Test Case Validation

// Generating sample KYC data

const customerKYCData = generateKYCData();

// Generating sample AML transaction data

const amlTransactionData = generateAMLTransaction();

// Testing KYC Validation

const kycValidationResult = validateKYC(customerKYCData);

console.log('KYC Validation Result:', kycValidationResult);

// Testing AML Transaction Validation

const amlValidationResult = validateAMLTransaction(amlTransactionData, customerKYCData);

console.log('AML Validation Result:', amlValidationResult);

// Test Case 1: Simulate large single transaction (greater than $50,000)

const largeTransaction = generateAMLTransaction();

largeTransaction.amount = '60000';  // Simulating a transaction > $50,000

const largeTransactionValidation = validateAMLTransaction(largeTransaction, customerKYCData);

console.log('Large Transaction Test Case:', largeTransactionValidation);

// Test Case 2: Simulate cross-border transaction

const crossBorderTransaction = generateAMLTransaction();

crossBorderTransaction.senderCountry = 'USA';

crossBorderTransaction.receiverCountry = 'Germany';

const crossBorderTransactionValidation = validateAMLTransaction(crossBorderTransaction, customerKYCData);

console.log('Cross Border Transaction Test Case:', crossBorderTransactionValidation);

// Test Case 3: Simulate rapid large transaction (multiple transactions)

const rapidTransaction = generateAMLTransaction();

rapidTransaction.senderAccountHistory = 1500;  // Simulating multiple large transactions in a short period

const rapidTransactionValidation = validateAMLTransaction(rapidTransaction, customerKYCData);

console.log('Rapid Large Transaction Test Case:', rapidTransactionValidation);

// Test Case 4: Suspicious source of funds from a high-risk country

const highRiskTransaction = generateAMLTransaction();

highRiskTransaction.senderCountry = 'North Korea';  // High-risk country

const highRiskTransactionValidation = validateAMLTransaction(highRiskTransaction, customerKYCData);

console.log('High-Risk Country Transaction Test Case:', highRiskTransactionValidation);

**Run the application**

