Contents

[1. Data creation using GPT: 1](#_Toc208631154)

[Prompt : 1](#_Toc208631155)

[Output: 2](#_Toc208631156)

[2. Prompt for Play Wright script to generate the Code – test data creation: 3](#_Toc208631157)

[Prompt: 3](#_Toc208631158)

[Code output: 4](#_Toc208631159)

[Code Result : 7](#_Toc208631160)

[3. LLM postman: 7](#_Toc208631161)

[Prompt: 7](#_Toc208631162)

[Output: 8](#_Toc208631163)

## Data creation using GPT:

### Prompt :

You are a strategic AI assistant expert in User acceptance Test.

Generate test data for credit card applications for different types of cards and conditions.

The goal is to generate test data for credit cards - different types like Master, Visa, Discovery etc and generate details on card number, type, expiry date, name, CVV

data should cover all edge cases boundary values null check ICE POT Framework for Credit Card Test Data Generation

using prompt first (using ICE POT)

Impact: 9 — Comprehensive test data ensures robust validation for credit card applications.

Confidence: 9 — Data generation logic is well-understood and repeatable.

Ease: 8 — Card details can be programmatically generated with realistic formats.

Potential: 9 — Enables thorough testing for multiple card types and edge cases.

Opportunity: 8 — Supports automation, negative testing, and compliance checks.

Time: 7 — Generation is quick with templated logic

### Output:

| **Card Type** | **Card Number** | **Expiry Date** | **Name on Card** | **CVV** | **Condition** |
| --- | --- | --- | --- | --- | --- |
| MasterCard | 5555 4444 3333 1111 | 12/27 | Alice Johnson | 123 | Valid |
| Visa | 4111 1111 1111 1111 | 09/26 | Bob Smith | 456 | Valid |
| Discover | 6011 0009 9013 9424 | 03/28 | Carol Lee | 789 | Valid |
| MasterCard | 5105 1051 0510 5100 | 01/20 | David Brown | 321 | Expired |
| Visa | 4000 0000 0000 0002 | 11/24 | Eve White | 654 | Invalid Number |
| Discover | 6011 1111 1111 1117 | 07/29 | Frank Green | 987 | Future Expiry |
| MasterCard | 2221 0000 0000 0009 | 05/30 | Grace Black | 159 | Valid |
| Visa | 4012 8888 8888 1881 | 08/28 | Henry Adams | 753 | Valid |
| Discover | 6011 6011 6011 6611 | 04/23 | Irene Clark | 852 | Expired |
| MasterCard | 2720 0000 0000 0007 | 10/27 | Jack Miller | 951 | Valid |
| Visa |  | 12/27 | Null Card | 123 | Null Card Number |
| Discover | 6011 0009 9013 9424 |  | Null Expiry | 789 | Null Expiry Date |
| MasterCard | 5555 4444 3333 1111 | 12/27 |  | 123 | Null Name |
| Visa | 4111 1111 1111 1111 | 09/26 | Bob Smith |  | Null CVV |
| Discover | 6011 0009 9013 9424 | 03/28 | Carol Lee | 0 | Boundary CVV (Min) |
| MasterCard | 5555 4444 3333 1111 | 12/99 | Max Expiry | 999 | Boundary Expiry (Max) |
| Visa | 4111 1111 1111 1111 | 01/00 | Min Expiry | 001 | Boundary Expiry (Min) |

## Prompt for Play Wright script to generate the Code – test data creation:

### Prompt:

You are a strategic AI assistant expert in Data Analyst who generates data.

Generate play wright script to generate test data for credit card applications for different types of cards and conditions.  
The goal is to generate test data for credit cards - different types like Master, Visa, Discovery etc and generate details on card number, type, expiry date, name, CVV  
data should cover all edge cases boundary values null check ICE POT Framework for Credit Card Test Data Generation  
using prompt first (using ICE POT)  
Impact: 9 — Comprehensive test data ensures robust validation for credit card applications.  
Confidence: 9 — Data generation logic is well-understood and repeatable.  
Ease: 8 — Card details can be programmatically generated with realistic formats.  
Potential: 9 — Enables thorough testing for multiple card types and edge cases.  
Opportunity: 8 — Supports automation, negative testing, and compliance checks.  
Time: 7 — Generation is quick with templated logic

### Code output:

import { test } from '@playwright/test';

/\*\*

 \* ICE POT Framework for Auto-Generating Credit Card Test Data

 \* I - Initialize card types and edge case conditions

 \* C - Create logic for generating realistic card details

 \* E - Execute generation for all combinations and edge cases

 \* P - Perform optional validation

 \* O - Output generated test data in table format

 \* T - Terminate (end of test)

 \*/

// Helper functions for card number, expiry, CVV, name

function generateCardNumber(type: string, condition: string): string {

  if (condition === 'Null Card Number') return '';

  if (condition === 'Invalid Number') return '1234 5678 9012 3456';

  switch (type) {

    case 'MasterCard': return '5555 4444 3333 1111';

    case 'Visa': return '4111 1111 1111 1111';

    case 'Discover': return '6011 0009 9013 9424';

    default: return '0000 0000 0000 0000';

  }

}

function generateExpiryDate(condition: string): string {

  if (condition === 'Null Expiry Date') return '';

  if (condition === 'Expired') return '01/20';

  if (condition === 'Future Expiry') return '12/99';

  if (condition === 'Boundary Expiry (Min)') return '01/00';

  if (condition === 'Boundary Expiry (Max)') return '12/99';

  return '12/27';

}

function generateName(condition: string): string {

  if (condition === 'Null Name') return '';

  return `${condition.replace(/[^a-zA-Z]/g, '')} User`;

}

function generateCVV(condition: string): string {

  if (condition === 'Null CVV') return '';

  if (condition === 'Boundary CVV (Min)') return '000';

  if (condition === 'Boundary CVV (Max)') return '999';

  if (condition === 'Expired') return '321';

  if (condition === 'Valid') return '123';

  return '111';

}

test('Auto-generate Credit Card Test Data (ICE POT)', async () => {

  // I: Initialize card types and conditions

  const cardTypes = ['MasterCard', 'Visa', 'Discover'];

  const conditions = [

    'Valid',

    'Expired',

    'Invalid Number',

    'Future Expiry',

    'Null Card Number',

    'Null Expiry Date',

    'Null Name',

    'Null CVV',

    'Boundary CVV (Min)',

    'Boundary CVV (Max)',

    'Boundary Expiry (Min)',

    'Boundary Expiry (Max)'

  ];

  // C/E: Generate test data for all combinations

  const testData = [];

  for (const type of cardTypes) {

    for (const condition of conditions) {

      testData.push({

        cardType: type,

        cardNumber: generateCardNumber(type, condition),

        expiryDate: generateExpiryDate(condition),

        nameOnCard: generateName(condition),

        cvv: generateCVV(condition),

        condition

      });

    }

  }

  // O: Output generated test data to console in table format

  console.table(testData);

});

### Code Result :

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

## LLM postman:

### Prompt:

{

   "model": "meta-llama/llama-4-scout-17b-16e-instruct",

  "max\_completion\_tokens": 1024,

  "top\_p": 0.95,

  "messages": [

    {

      "role": "system",

      "content": "You are a strategic AI assistant expert in User acceptance Test."

    },

    {

      "role": "user",

      "content": "generate test data for credit card applications for different types of cards and conditions. The goal is to generate test data for credit cards - different types like Master, Visa, Discovery etc and generate details on card number, type, expiry date, name, CVV data should cover all edge cases boundary values null check ICE POT Framework for Credit Card Test Data Generation using prompt first (using ICE POT)Impact: 9 — Comprehensive test data ensures robust validation for credit card applications. Confidence: 9 — Data generation logic is well-understood and repeatable. Ease: 8 — Card details can be programmatically generated with realistic formats. Potential: 9 — Enables thorough testing for multiple card types and edge cases. Opportunity: 8 — Supports automation, negative testing, and compliance checks. Time: 7 — Generation is quick with templated logic"

    },

    {

      "role": "assistant",

      "content": " Output only credit card data in table format"

    }

  ]

}

### Output:

{

    "id": "chatcmpl-6fd199ba-0222-407e-8437-55c5abf4da70",

    "object": "chat.completion",

    "created": 1757718136,

    "model": "meta-llama/llama-4-scout-17b-16e-instruct",

    "choices": [

        {

            "index": 0,

            "message": {

                "role": "assistant",

                "content": ":\n\n| \*\*Card Type\*\* | \*\*Card Number\*\* | \*\*Expiry Date\*\* | \*\*Name\*\* | \*\*CVV\*\* |\n| --- | --- | --- | --- | --- |\n| Master | 5105105105105100 | 12/2025 | John Doe | 123 |\n| Visa | 4111111111111111 | 06/2028 | Jane Smith | 456 |\n| Discover | 6011111111111117 | 09/2026 | Bob Johnson | 789 |\n| Amex | 371449635398431 | 03/2027 | Alice Brown | 901 |\n| Diners Club | 30569309025904 | 11/2024 | Mike Davis | 234 |\n| JCB | 3530111333300000 | 08/2029 | Emily Taylor | 567 |\n| Master |  |  |  |  |\n| Visa |  |  |  |  |\n| Discover |  |  |  |  |\n\nICE POT Framework for Credit Card Test Data Generation:\n\n### \*\*I\*\* - \*\*Input\*\* \n\n\*   Card Type (Master, Visa, Discover, Amex, Diners Club, JCB)\n\*   Card Holder Name \n\*   Expiry Date \n\*   CVV \n\n### \*\*C\*\* - \*\*Constraints\*\* \n\n\*   Card Number: 13-16 digits, Luhn algorithm \n\*   Expiry Date: MM/YYYY, future date \n\*   CVV: 3-4 digits \n\n### \*\*E\*\* - \*\*Edge\*\* \n\n\*   Null/Empty card number \n\*   Invalid card number (less/more than 16 digits) \n\*   Expired card \n\*   CVV with less/more than 3 digits \n\n### \*\*P\*\* - \*\*Positive\*\* \n\n\*   Valid card details \n\*   Different card types \n\*   Various expiry dates \n\n### \*\*O\*\* - \*\*Opportunity\*\* \n\n\*   \*\*T\*\* - \*\*Test\*\* \n\n    \*   \*\*Data\*\* \n\n    \*   \*\*Conditions\*\* \n\n    \*   \*\*Generation\*\* \n\nExample Python code to generate test data:\n\n```python\nimport random\nimport datetime\n\ndef generate\_card\_data(card\_type):\n    card\_types = {\n        'Master': '5105105105105100',\n        'Visa': '4111111111111111',\n        'Discover': '6011111111111117',\n        'Amex': '371449635398431',\n        'Diners Club': '30569309025904',\n        'JCB': '3530111333300000'\n    }\n\n    card\_number = card\_types[card\_type]\n    expiry\_date = (datetime.date.today() + datetime.timedelta(days=random.randint(1, 365\*2))).strftime('%m/%Y')\n    name = f'Customer {card\_type}'\n    cvv = str(random.randint(100, 999))\n\n    return {\n        'Card Type': card\_type,\n        'Card Number': card\_number,\n        'Expiry Date': expiry\_date,\n        'Name': name,\n        'CVV': cvv\n    }\n\n# Generate test data for different card types\ncard\_types = ['Master', 'Visa', 'Discover', 'Amex', 'Diners Club', 'JCB']\n\nfor card\_type in card\_types:\n    print(generate\_card\_data(card\_type))\n```\n\nThe code example provided generates test data covering the following scenarios:\n\n\*   \*\*Valid card details\*\* for different card types \n\*   \*\*Expiry dates\*\* in the future \n\*   \*\*CVV\*\* values with 3 digits \n\nThe data generation can be extended to cover more edge cases, such as:\n\n\*   \*\*Null/empty card number\*\* \n\*   \*\*Invalid card numbers\*\* (e.g., less/more than 16 digits) \n\*   \*\*Expired cards\*\* \n\nBy using the \*\*ICE POT\*\* framework and the provided Python code, you can efficiently generate comprehensive test data for credit card applications, ensuring robust validation and thorough testing for multiple card types and edge cases."

            },

            "logprobs": **null**,

            "finish\_reason": "stop"

        }

    ],

    "usage": {

        "queue\_time": 0.07161258,

        "prompt\_tokens": 195,

        "prompt\_time": 0.00526576,

        "completion\_tokens": 833,

        "completion\_time": 1.911608662,

        "total\_tokens": 1028,

        "total\_time": 1.916874422

    },

    "usage\_breakdown": **null**,

    "system\_fingerprint": "fp\_5d3e4e58e1",

    "x\_groq": {

        "id": "req\_01k502mfsjfp8rsqfv210nbtr2"

    },

    "service\_tier": "on\_demand"

}ssssss