LLM-Based Test Case Generation System

Complete Technical Manual

Version: 1.0 **Date:** January 2025 **Status:** Proof of Concept (POC)

Table of Contents

1. Executive Summary

What Is This?

A system that uses Large Language Models (GPT-4o) to automatically generate executable test cases for complex web forms. Instead of manually writing hundreds of CSV rows, testers describe scenarios in plain English and receive ready-to-use automation scripts.

Key Innovation

Traditional: Human writes 150+ row CSV manually for each test scenario **New:** Human writes one sentence, AI generates the complete CSV

Business Value

Metric	Before	After
Time per test case	2-4 hours	30 seconds
Cost per test case	~$100 (labor)	~$0.06 (API)

Metric	Before	After
Possible coverage	10-20 scenarios	Unlimited
Human error rate	High	Validated

2. The Problem We Solved

The Onboarding Form Challenge

Our target application is a credit union member onboarding form with:

```
Form Structure:
├── Page 1: Contact Info (name, email, phone, OTP)
├── Page 2: Documents (ID uploads, utility bills, address)
├── Page 3: Additional Details (employment, income, nationality)
├── Page 4: Other Products (beneficiaries, joint partners, LinCU, FIP)
├── Page 5: PEP/FATCA (16 compliance questions)
└── Page 6: PDF Review & Final Submission
```

The Combinatorial Explosion

Field Type	Options	Combinations
Binary Yes/No fields	15+ fields	2^15 = 32,768
Employment Status	8 options	× 8
Marital Status	6 options	× 6
Salary Range	6 options	× 6
Beneficiaries	0-8 people	× 9
Joint Partners	0-3 people	× 4
Dependents	0-10 people	× 11

Total Possible Paths: Millions of unique test scenarios

Why Manual Testing Fails

1. **Volume:** Impossible to write test cases for all combinations
2. **Time:** Each CSV has 150-400 rows requiring precise XPaths
3. **Accuracy:** One wrong XPath = test failure
4. **Maintenance:** Form changes = rewrite all test cases
5. **Conditional Logic:** Human must track "if X then Y" rules mentally

3. Solution Overview

The Core Idea

Use an LLM as a "test case composer" that:

1. **Understands** the complete form structure from a JSON schema
2. **Knows** all conditional business rules (if X, show Y)
3. **Generates** specific test cases from natural language descriptions
4. **Outputs** executable CSV files for Selenium/Playwright

Why LLMs Work Here

LLM Strength	Application
Context understanding	Comprehends 16K tokens of schema + rules
Reasoning	Applies conditional logic correctly
Pattern following	Matches exact CSV format from examples
Natural language	Interprets human scenario descriptions

The Trade-Off

We trade **compute cost** (~$0.06/call) for **human time** (hours saved per test case).

---

4. System Architecture

High-Level Architecture

```
┌──────────────────────────────────────────────────────────────────────┐
│                          USER INPUT                                    │
│  "Single male, full-time employed, no beneficiaries, all PEP = No"   │
└──────────────────────────────────────────────────────────────────────┘
                                   │
                                   ▼
┌──────────────────────────────────────────────────────────────────────┐
│                       CONTEXT ASSEMBLY                                 │
│                                                                     │
│  ┌────────────────┐  ┌───────────────────┐  ┌────────────────────┐   │
│  │ form_schema    │  │ conditional_rules │  │ example_csv         │  │
│  │   .json        │  │      .json        │  │                     │  │
│  │  (~8.8K tok)   │  │   (~1.7K tok)     │  │  (~4.6K tok)        │  │
│  └────────────────┘  └───────────────────┘  └────────────────────┘   │
│                                  │                                     │
│                                  ▼                                     │
│                     Total Context: ~16,000 tokens                      │
└──────────────────────────────────────────────────────────────────────┘
                                   │
                                   ▼
┌──────────────────────────────────────────────────────────────────────┐
│                           GPT-4o API                                   │
│                                                                        │
│  • Receives: System prompt (schema + rules + example) + User query │
│  • Processes: Applies conditional logic, generates CSV             │
│  • Returns: Raw CSV content (~2,000 tokens)                        │
│  • Cost: ~$0.09 per call                                           │
└────────────────────────────────────────────────────────────────────┘
                                │
                                ▼
┌────────────────────────────────────────────────────────────────────┐
│                     POST-PROCESSING                                 │
│                                                                     │
│  fix_csv_quoting()                                                 │
│  • Detects unquoted comma-containing values                        │
│  • Rewrites CSV with proper quoting                                │
│  • Handles edge cases like "$12,001-$17,000"                       │
└────────────────────────────────────────────────────────────────────┘
                                │
                                ▼
┌────────────────────────────────────────────────────────────────────┐
│                      VALIDATION                                     │
│                                                                     │
│  ┌─────────────────────┐   ┌─────────────────────────────────┐    │
│  │  validator.py       │   │  scenario_validator.py          │    │
│  │  (Structural)       │   │  (Semantic)                     │    │
│  │                     │   │                                 │    │
│  │  • 7 columns?       │   │  • Employment matches prompt?   │    │
│  │  • Valid XPaths?    │   │  • Beneficiary setting correct? │    │
│  │  • Page order?      │   │  • PEP/FATCA all No?            │    │
│  │  • Actions valid?   │   │  • LinCU/FIP as requested?      │    │
│  └─────────────────────┘   └─────────────────────────────────┘    │
└────────────────────────────────────────────────────────────────────┘
                                │
                                ▼
┌────────────────────────────────────────────────────────────────────┐
│                       OUTPUT                                        │
│                                                                     │
│  generated/single_male_fulltime_20250131_143022.csv                │
│  • ~150-400 rows depending on scenario complexity                  │
│  • Ready for Selenium/Playwright execution                         │
└────────────────────────────────────────────────────────────────────┘
```

Component Interaction

```
┌─────────────────────────────────────────────────────────────┐
│                    generate_test_case.py                      │
│                       (Orchestrator)                          │
├─────────────────────────────────────────────────────────────┤
│                                                             │
│   INPUTS                     PROCESS                OUTPUTS │
│   ───────                    ───────                ─────── │
│                                                             │
│   form_schema.json ──┐                                      │
│                       │                                   │
│   conditional_        ├───►┌─────────────┐                │
│   rules.json    ──────┤    │ Build System │               │
│                       │    │   Prompt    │                │
│   example.csv   ──────┘    └──────┬──────┘                │
│                                   │                       │
│                                   ▼                       │
│                                                           │
│   User Scenario ────────►┌─────────────┐                  │
│                          │ GPT-4o API  │                  │
│                          └──────┬──────┘                  │
│                                 │                         │
│                                 ▼                         │
│                          ┌─────────────┐    ┌──────────┐  │
│                          │ Fix CSV     │───►│ Save CSV │  │
│                          │ Quoting     │    └────┬─────┘  │
│                          └─────────────┘         │        │
│                                                  ▼        │
│                                      generated/*.csv      │
│                                      cost_log.json        │
└───────────────────────────────────────────────────────────┘
```

5. Data Flow

Complete Data Journey

```
PHASE 1: CONTEXT PREPARATION
════════════════════════════

form_schema.json                    conditional_rules.json
      │                                       │
      │ Contains:                             │ Contains:
      │ • 6 pages definition                  │ • 12 dependency rules
      │ • All field IDs & types               │ • Trigger conditions
      │ • Dropdown options                    │ • Required/excluded fields
      │ • XPath locators                      │ • Repeatable section limits
      │ • Validation rules                    │
      │                                       │
      └───────────────┬───────────────────────┘
                      │
                      ▼
              System Prompt
              (~16,000 tokens)
                      │
                      │ + example.csv (format reference)
                      │
                      ▼

PHASE 2: GENERATION
═══════════════════

User Input: "Single male, full-time, no beneficiaries"
```

```
                     │
                     ▼
              ┌─────────────┐
              │   GPT-4o    │
              │             │
              │ Reasoning:  │
              │ 1. Parse requirements
              │ 2. Apply Rule R003 (employed → employer fields)
              │ 3. Apply Rule R006 (no beneficiary → skip section)
              │ 4. Generate all pages in order
              │ 5. Use exact XPaths from schema
              └──────┬──────┘
                     │
                     ▼
              Raw CSV Output
              (~150 rows)


PHASE 3: POST-PROCESSING
════════════════════════

Raw CSV ──► fix_csv_quoting() ──► Clean CSV
                   │
                   │ Fixes:
                   │ • Unquoted commas in values
                   │ • Malformed fields
                   │ • Column count mismatches


PHASE 4: VALIDATION
═══════════════════

Clean CSV
    │
    ├──► validator.py (Structure)
    │         │
    │         └──► ✓ Columns exist
    │              ✓ XPaths valid
    │              ✓ Page order correct
    │
    └──► scenario_validator.py (Semantics)
              │
              └──► ✓ Employment matches
                   ✓ Beneficiary setting correct
                   ✓ PEP/FATCA as requested


PHASE 5: OUTPUT
═══════════════

generated/
├── scenario_name_20250131_143022.csv  ◄── Ready for automation
└── cost_log.json                       ◄── API cost tracking
```

Data Transformation Example

Input (Natural Language):

```
"Single male, full-time employed at AGOSTINI'S, no beneficiaries"
```

Intermediate (AI Reasoning):

```
Requirements extracted:
- marital_status = SINGLE
- employment_status = FULL TIME PERMANENT
- employer = AGOSTINI'S LIMITED
- has_beneficiary = false

Rules applied:
- R003: Include employer, occupation, workPhoneNo (employed)
- R006: Skip beneficiary fields (hasBeneficiary = false)
```

Output (CSV rows - excerpt):

```
,Group,Element,Action,Value,Strategy,XPath
0,Contact Info,firstName,click,,id,"//*[@id=""firstName""]"
1,Contact Info,firstName,Input,JOHN,id,"//*[@id=""firstName""]"
...
45,Additional Details,Employment Status,click,,data-testid,"..."
46,Additional Details,FULL TIME PERMANENT,click,,data-testid,"//*[@data-testid=""option-0""]"
47,Additional Details,employer,click,,data-testid,"..."
48,Additional Details,AGOSTINI'S LIMITED,click,,data-testid,"//*[@data-testid=""option-3""]"
...
90,Other Products,hasBeneficiary,click,,absolute,"/html/.../label[2]/input[1]"
   ↑ No beneficiary fields follow because hasBeneficiary = false
```

---

6. File Structure & Purpose

```
version4.1/schema/
│
├── 🔵 CORE DATA FILES
│   │
│   ├── form_schema.json
│   │   Purpose: Complete form structure definition
│   │   Contains: Pages, sections, fields, XPaths, dropdown options
│   │   Size: ~900 lines, ~8,800 tokens
│   │   Used by: generate_test_case.py (context for LLM)
│   │
│   └── conditional_rules.json
│       Purpose: Business logic rules
│       Contains: 12 dependency rules, repeatable section limits
│       Size: ~260 lines, ~1,700 tokens
│       Used by: generate_test_case.py (context for LLM)
│
├── 🟢 MAIN SCRIPTS
│   │
│   ├── generate_test_case.py
│   │   Purpose: Orchestrates the entire generation flow
│   │   Functions:
│   │   - load_json(): Load schema and rules
│   │   - build_system_prompt(): Assemble LLM context
│   │   - generate_test_case(): Call GPT-4o API
│   │   - fix_csv_quoting(): Post-process CSV
│   │   - save_test_case(): Write output file
│   │   - Cost tracking via SessionStats
│   │
│   ├── validator.py
│   │   Purpose: Structural CSV validation
│   │   Checks: Column count, XPath format, page order, action types
│   │
│   └── scenario_validator.py
│       Purpose: Semantic validation
│       Checks: Does CSV match the scenario requirements?
│       Features:
│       - Natural language parsing
│       - Boolean detection from XPath patterns
│       - Requirement matching
│
├── 📋 REFERENCE DATA
│   │
│   └── examples/
│       └── simple_flow_example.csv
│           Purpose: Few-shot learning example for LLM
│           Contains: 154-row complete test case
│           Used by: build_system_prompt() as format reference
│
├── 📦 OUTPUT
│   │
│   └── generated/
│       ├── *.csv                  Generated test cases
│       └── cost_log.json          API cost tracking
│
├── 📚 DOCUMENTATION
│   │
│   ├── README.md                  Quick start guide
│   ├── ARCHITECTURE.md            Technical architecture
│   ├── manual.md                  This comprehensive manual
│   └── prompt_engineering_strategy.md
│                                  Prompting guidelines
│
```

```
│
└── 📦 DEPENDENCIES
    │
    ├── requirements.txt           Python packages
    │   - openai>=1.0.0
    │   - python-dotenv>=1.0.0
    │
    └── ../.env                    API keys (parent directory)
        - OPENAI_API_KEY
```

7. Core Components Deep Dive

7.1 form_schema.json

Purpose: Provides complete structural knowledge of the form to the LLM.

Hierarchy:

```
form_schema.json
├── pages[]                    # 6 pages
│   ├── id                     # Unique identifier
│   ├── name                   # Display name (used in CSV "Group" column)
│   ├── order                  # Page sequence (1-6)
│   ├── sections[]             # Logical groupings within page
│   │   ├── id
│   │   ├── name
│   │   ├── repeatable         # true for beneficiaries, etc.
│   │   ├── min/max_instances  # Limits for repeatable sections
│   │   └── fields[]           # Individual form elements
│   │       ├── id             # Field identifier
│   │       ├── type           # text, dropdown, boolean, file, otp,
button
│   │       ├── required       # Is field mandatory?
│   │       ├── strategy       # Locator strategy (id, data-testid,
absolute)
│   │       ├── xpath          # Primary XPath
│   │       ├── xpath_trigger  # For dropdowns: click to open
│   │       ├── xpath_true     # For booleans: click for Yes
│   │       ├── xpath_false    # For booleans: click for No
│   │       ├── options[]      # For dropdowns: available choices
│   │       │   ├── value      # Display text
│   │       │   └── xpath      # XPath to select this option
│   │       └── conditional    # When is this field visible?
│   └── navigation             # Save/Continue buttons
└── global_rules               # Defaults (OTP value, country, files)
```

Example Field Types:

```
// Text field
{
  "id": "firstName",
  "type": "text",
  "required": true,
  "strategy": "id",
  "xpath": "//*[@id=\"firstName\"]",
  "example_values": ["JOHN", "MARIA"]
}

// Dropdown field
{
  "id": "employmentStatus",
  "type": "dropdown",
  "xpath_trigger": "//*[@data-testid=\"dropdown-text\"...]",
  "options": [
    {"value": "FULL TIME PERMANENT", "xpath": "//*[@data-testid=\"option-0\"]"},
    {"value": "UNEMPLOYED", "xpath": "//*[@data-testid=\"option-7\"]"}
  ]
}

// Boolean field
{
  "id": "hasBeneficiary",
  "type": "boolean",
  "xpath_true": "/html/.../label[1]/input[1]",
  "xpath_false": "/html/.../label[2]/input[1]"
}
```

7.2 conditional_rules.json

Purpose: Encodes all business logic so LLM knows which fields to include/exclude.

Rule Structure:

```
{
  "rule_id": "R003",
  "name": "Employed Status Fields",
  "trigger_field": "employmentStatus",
  "trigger_values": ["FULL TIME PERMANENT", "FULL TIME TEMPORARY", "PART TIME"],
  "required_fields": ["employer", "occupation", "workPhoneNo", ...],
  "excluded_fields": []
}
```

Key Rules:

Rule	Trigger	Effect

Rule	Trigger	Effect
R001	permanentAddressSameAsMailing = false	Show permanent address fields
R003	employmentStatus = EMPLOYED variants	Show employer fields
R004	employmentStatus = SELF EMPLOYED	Show self-employed fields, hide employer
R005	employmentStatus = RETIRED/UNEMPLOYED	Hide all employer fields
R006	hasBeneficiary = true	Show beneficiary section
R007	hasJointPartner = true	Show joint partner section
R008	isApplyingForFipApplication = true	Show FIP plan selection

7.3 generate_test_case.py

Key Functions:

```
# 1. Load context files
schema = load_json("form_schema.json")
rules = load_json("conditional_rules.json")
example_csv = load_file("examples/simple_flow_example.csv")

# 2. Build system prompt (~16K tokens)
system_prompt = build_system_prompt(schema, rules, example_csv)

# 3. Call GPT-4o API
response, usage = generate_test_case(
    client=openai_client,
    system_prompt=system_prompt,
    user_scenario="Single male, full-time employed...",
    model="gpt-4o"
)

# 4. Fix CSV quoting issues
fixed_csv = fix_csv_quoting(response)

# 5. Save and track costs
filepath = save_test_case(fixed_csv, scenario_name)
save_cost_log(session, output_dir)
```

The fix_csv_quoting() Function:

Problem: LLM sometimes outputs `$12,001-$17,000` without quotes, breaking CSV parsing.

Solution:

1. Parse each line character by character
2. Track quote state to handle embedded commas
3. Identify the Action column (always "click" or "Input")

4. Reconstruct with proper quoting

7.4 scenario_validator.py

Purpose: Verify generated CSV matches the scenario requirements.

Key Innovation - Boolean Detection from XPath:

The form uses radio buttons where:

- `label[1]` = Yes/True
- `label[2]` = No/False

```
def get_boolean_value(self, element_name: str) -> Optional[bool]:
    # Check click action XPath
    for row in self.rows:
        if row.get('Element') == element_name and row.get('Action') ==
'click':
            xpath = row.get('XPath', '')
            if 'label[1]' in xpath:
                return True
            elif 'label[2]' in xpath:
                return False
    return None
```

Validation Checks:

Check	Method	Pass Criteria
Employment Status	Dropdown selection	Matches prompt
Marital Status	Dropdown selection	Matches prompt
Has Beneficiary	Boolean from XPath	Matches prompt
LinCU Card	Boolean from XPath	Matches prompt
FIP Application	Boolean from XPath	Matches prompt
All PEP = No	All 11 booleans	All False
All FATCA = No	All 5 booleans	All False
Page Coverage	Group column	All 6 pages present
OTP Verifications	Element count	>= 2 instances

8. How It Works: Step-by-Step

Complete Walkthrough

Step 1: User Runs Command

```
cd /Users/impactoinfra/test_case_generation/version4.1/schema
python generate_test_case.py "Single male, full-time employed at
AGOSTINI'S,
no beneficiaries, no joint partners, no LinCU, no FIP, all PEP/FATCA = No"
```

Step 2: Script Loads Context

```
Loading form_schema.json...     ✓ (~8,800 tokens)
Loading conditional_rules.json... ✓ (~1,700 tokens)
Loading example CSV...          ✓ (~4,600 tokens)
Building system prompt...       ✓ (~16,000 tokens total)
```

Step 3: API Call to GPT-4o

```
Sending request to OpenAI API...
Model: gpt-4o
Temperature: 0.2 (low for consistency)
Max tokens: 16,000
```

Step 4: LLM Processes Request

The LLM internally:

1. Parses "single male" → maritalStatus = SINGLE
2. Parses "full-time employed" → employmentStatus = FULL TIME PERMANENT
3. Looks up Rule R003 → Must include employer fields
4. Parses "no beneficiaries" → hasBeneficiary = false
5. Looks up Rule R006 → Skip all beneficiary fields
6. Generates CSV following exact page order
7. Uses XPaths from schema, not invented

Step 5: Post-Processing

```
Received response (2,156 tokens)
Fixing CSV quoting issues...
Saving to generated/single_male_fulltime_20250131_143022.csv
```

Step 6: Cost Logging

```
============================================================
                    USAGE STATISTICS
============================================================
Prompt tokens:     16,234    Cost: $0.0406
Completion tokens: 2,156     Cost: $0.0216
```

```
Total cost:        $0.0621
============================================================
```

Step 7: Validation (Optional)

```
python scenario_validator.py generated/test.csv "Single male, full-time..."

======================================================================
SCENARIO VALIDATION REPORT
======================================================================
✅ PASS Employment Status    Expected: FULL TIME PERMANENT  Actual: FULL TIME PERMANENT
✅ PASS Marital Status       Expected: SINGLE               Actual: SINGLE
✅ PASS Has Beneficiary      Expected: False                Actual: False
✅ PASS LinCU Card           Expected: False                Actual: False
✅ PASS All PEP = No         Expected: All False            Actual: All False
======================================================================
RESULT: ✅ VALIDATION PASSED
======================================================================
```

---

9. Use Case Examples

Example 1: Simple Flow (Minimal Options)

Scenario:

```
"Single male, full-time permanent employed, no beneficiaries, no joint partners,
no LinCU card, no FIP application, all PEP/FATCA questions = No"
```

Expected CSV Characteristics:

- ~150 rows
- Employment section: Full employer fields
- Other Products: All toggles set to No/False
- PEP/FATCA: All 16 questions = No

Command:

```
python generate_test_case.py "Single male, full-time permanent, no beneficiaries,
no LinCU, no FIP, all PEP/FATCA = No"
```

Example 2: Complex Flow (Multiple Beneficiaries)

Scenario:

```
"Married female, self-employed, 3 beneficiaries (each 33%), applying for
LinCU card,
FIP Plan B, all PEP/FATCA = No"
```

Expected CSV Characteristics:

- ~300+ rows
- Employment: Self-employed fields (no employer)
- Beneficiaries: 3 complete instances with:
    - Document type selection
    - Mobile number
    - Relation dropdown
    - ID upload
    - Percentage (33%, 33%, 34%)
- LinCU: Toggle = Yes
- FIP: Toggle = Yes, Plan = B

Command:

```
python generate_test_case.py "Married female, self-employed, 3
beneficiaries,
LinCU card yes, FIP Plan B, all PEP/FATCA = No"
```

Example 3: Edge Case (Unemployed with Joint Partner)

Scenario:

```
"Divorced male, unemployed, no beneficiaries, 1 joint partner,
no LinCU, no FIP, all PEP/FATCA = No"
```

Expected CSV Characteristics:

- Employment section: Minimal (no employer fields)
- Joint Partner: 1 instance with member ID search
- Salary: May still require selection (even if unemployed)

Command:

```
python generate_test_case.py "Divorced male, unemployed, 1 joint partner,
no LinCU, no FIP, all PEP/FATCA = No"
```

Example 4: PEP Positive Scenario

Scenario:

```
"Single male, full-time employed, no beneficiaries, isHeadOfGovt = Yes,
all other PEP = No, all FATCA = No"
```

Expected CSV Characteristics:

- PEP section: isHeadOfGovt uses label[1] XPath (Yes)
- All other PEP fields use label[2] XPath (No)
- May trigger additional documentation requirements

---

10. Comparison: LLM vs Traditional Approach

Traditional Manual Approach

```
┌─────────────────────────────────────────────────────────────────┐
│                      MANUAL PROCESS                             │
├─────────────────────────────────────────────────────────────────┤
│                                                                 │
│  1. QA Engineer reads requirements                              │
│                    ↓                                            │
│  2. Opens form in browser, clicks through manually              │
│                    ↓                                            │
│  3. Records each element's XPath using DevTools                 │
│                    ↓                                            │
│  4. Creates spreadsheet with 150+ rows                          │
│                    ↓                                            │
│  5. Reviews for conditional logic errors                        │
│                    ↓                                            │
│  6. Tests CSV in automation framework                           │
│                    ↓                                            │
│  7. Debugs XPath errors (repeat steps 3-6)                      │
│                    ↓                                            │
│  8. Final test case ready                                       │
│                                                                 │
│  Time: 2-4 hours per test case                                  │
│  Error Rate: High (typos, wrong XPaths, missed conditions)      │
│                                                                 │
└─────────────────────────────────────────────────────────────────┘
```

LLM-Based Approach

```
┌─────────────────────────────────────────────────────────────────┐
│                      LLM PROCESS                                │
├─────────────────────────────────────────────────────────────────┤
│                                                                 │
│  1. QA Engineer describes scenario in English                   │
│                     ↓                                           │
│  2. System sends to GPT-4o with full context                    │
│                     ↓                                           │
│  3. LLM generates complete CSV                                  │
│                     ↓                                           │
│  4. Auto-fix CSV quoting issues                                 │
│                     ↓                                           │
│  5. Validate against requirements                               │
│                     ↓                                           │
│  6. Test case ready                                             │
│                                                                 │
│  Time: 30-60 seconds per test case                              │
│  Error Rate: Low (validated, consistent XPaths)                 │
│                                                                 │
└─────────────────────────────────────────────────────────────────┘
```

Side-by-Side Comparison

Aspect	Manual	LLM-Based
XPath accuracy	Varies by engineer	Consistent from schema
Conditional logic	Mental tracking	Encoded in rules
Scalability	Linear (more engineers)	Near-infinite
Maintenance	Edit each CSV	Update schema once
Reproducibility	Low	High
Coverage possible	10-50 scenarios	Unlimited

When Manual is Still Needed

1. **Initial schema creation** - One-time effort to document form
2. **XPath extraction** - DevTools still needed to find locators
3. **Edge case debugging** - When LLM produces invalid output
4. **New form sections** - Schema must be updated first

---

11. Cost Analysis

GPT-4o Pricing (January 2025)

Token Type	Price
Input (prompt)	$2.50 / 1M tokens
Output (completion)	$10.00 / 1M tokens

Typical Test Case Generation Cost

```
CONTEXT (Input):
├── form_schema.json      ~8,800 tokens
├── conditional_rules.json ~1,700 tokens
├── example_csv           ~4,600 tokens
├── System instructions   ~1,100 tokens
└── User scenario         ~100 tokens
────────────────────────────────────
Total Input:              ~16,300 tokens
Cost:                     $0.041

OUTPUT (Completion):
├── Generated CSV         ~2,000 tokens
────────────────────────────────────
Total Output:             ~2,000 tokens
Cost:                     $0.020

TOTAL PER CALL:           ~$0.06
```

Batch Generation Economics

Scenarios	LLM Cost
10	$0.60
50	$3.00
100	$6.00
1,000	$60.00

12. Validation System

Two-Stage Validation

```
┌─────────────────────────────────────────────────────────────────┐
│                    STAGE 1: STRUCTURAL                          │
│                    (validator.py)                               │
├─────────────────────────────────────────────────────────────────┤
│                                                                 │
│  Checks:                                                        │
│  ✓ CSV has exactly 7 columns                                    │
│  ✓ Column names: (index), Group, Element, Action, Value,        │
│                  Strategy, XPath                                │
│  ✓ Action values are "click" or "Input" only                    │
│  ✓ XPath values start with "/" or "//"                          │
│  ✓ Pages appear in correct order (1→2→3→4→5→6)                  │
│  ✓ No empty required fields                                     │
│                                                                 │
│  Command:                                                       │
│  python validator.py test.csv form_schema.json conditional_rules.json
│                                                                 │
└─────────────────────────────────────────────────────────────────┘
                              │
                              ▼
┌─────────────────────────────────────────────────────────────────┐
│                    STAGE 2: SEMANTIC                            │
│                    (scenario_validator.py)                      │
├─────────────────────────────────────────────────────────────────┤
│                                                                 │
│  Process:                                                       │
│  1. Parse natural language scenario                             │
│     "Single male, full-time, no beneficiaries"                  │
│                    ↓                                            │
│  2. Extract structured requirements                             │
│     marital_status = SINGLE                                     │
│     employment_status = FULL TIME PERMANENT                     │
│     has_beneficiary = False                                     │
│                    ↓                                            │
│  3. Check each requirement against CSV                          │
│     - Find employmentStatus dropdown selection                  │
│     - Find hasBeneficiary boolean from XPath                    │
│     - Verify all match                                          │
│                                                                 │
│  Command:                                                       │
│  python scenario_validator.py test.csv "scenario description"   │
│                                                                 │
└─────────────────────────────────────────────────────────────────┘
```

Boolean Detection Logic

The Challenge: CSV often only shows `click` action, not explicit `true`/`false` value.

```
90,Other
Products,hasBeneficiary,click,,absolute,"/html/.../label[2]/input[1]"
```

The Solution: Detect boolean from XPath pattern:

- `label[1]` in XPath → True/Yes
- `label[2]` in XPath → False/No

This is specific to this form's radio button implementation.

Validation Report Example

```
======================================================================
SCENARIO VALIDATION REPORT
======================================================================
CSV File: generated/test_20250131_143022.csv
Scenario: Single male, full-time employed, no beneficiaries, no LinCU...
======================================================================

Summary: 8 passed, 0 failed, 1 warnings

ℹ️ INFO Row Count
      Expected: 100-400 typical
      Actual:   154
      Details:  Total rows in generated CSV

✅ PASS Page Coverage
      Expected: All 6 pages
      Actual:   6/6 pages
      Details:  All pages covered

✅ PASS Employment Status
      Expected: FULL TIME PERMANENT
      Actual:   FULL TIME PERMANENT

✅ PASS Marital Status
      Expected: SINGLE
      Actual:   SINGLE

✅ PASS Has Beneficiary
      Expected: False
      Actual:   False

✅ PASS LinCU Card Application
      Expected: False
      Actual:   False

✅ PASS FIP Application
      Expected: False
      Actual:   False

✅ PASS All PEP Questions = No
      Expected: All False
      Actual:   All False

✅ PASS All FATCA Questions = No
      Expected: All False
      Actual:   All False

⚠ WARN OTP Verifications
      Expected: 2 (initial + final)
      Actual:   4
```

```
======================================================================
RESULT: ✅ VALIDATION PASSED
======================================================================
```

13. Quick Reference & Commands

Setup

```bash
# Navigate to schema directory
cd /Users/impactoinfra/test_case_generation/version4.1/schema

# Install dependencies
pip install -r requirements.txt

# Set API key (in parent .env file)
echo "OPENAI_API_KEY=sk-..." > ../.env
```

Generation Commands

```bash
# Basic generation
python generate_test_case.py "your scenario description"

# Interactive mode
python generate_test_case.py -i

# Use cheaper model
python generate_test_case.py -m gpt-4o-mini "your scenario"

# Cost estimate only (no API call)
python generate_test_case.py --estimate-only
```

Validation Commands

```bash
# Structural validation
python validator.py generated/test.csv form_schema.json
conditional_rules.json

# Semantic validation
python scenario_validator.py generated/test.csv "your scenario
description"
```

Scenario Description Syntax

Requirement	Syntax Examples
Employment	"full-time permanent", "self-employed", "unemployed", "retired pensioned"
Marital Status	"single", "married", "divorced", "widowed"
Gender	"male", "female"
Beneficiaries	"no beneficiaries", "2 beneficiaries", "3 beneficiaries"
Joint Partners	"no joint partners", "1 joint partner"
LinCU Card	"no LinCU", "LinCU card yes", "applying for LinCU"
FIP	"no FIP", "FIP Plan A", "FIP Plan B"
PEP/FATCA	"all PEP/FATCA = No", "all PEP no, all FATCA no"

Example Scenarios

```
# Minimal flow
python generate_test_case.py "Single male, full-time permanent, no
beneficiaries,
no joint partners, no LinCU, no FIP, all PEP/FATCA = No"

# With beneficiaries
python generate_test_case.py "Married female, part-time employed, 2
beneficiaries,
no joint partners, LinCU yes, no FIP, all PEP/FATCA = No"

# Self-employed with FIP
python generate_test_case.py "Single male, self-employed, no
beneficiaries,
no joint partners, no LinCU, FIP Plan B, all PEP/FATCA = No"

# Complex flow
python generate_test_case.py "Divorced female, retired pensioned, 3
beneficiaries,
1 joint partner, LinCU yes, FIP Plan A, all PEP/FATCA = No"
```

---

14. Troubleshooting Guide

Common Issues

Issue 1: CSV Parsing Error

```
pandas.errors.ParserError: Expected 7 fields in line 62, saw 9
```

Cause: Value contains unquoted commas (e.g., `$12,001-$17,000`)

Solution: The `fix_csv_quoting()` function should handle this automatically. If not:

1. Check the raw CSV output
2. Manually quote the problematic value
3. Report to improve the fix function

Issue 2: Validation Fails on Boolean Field

```
❌ FAIL Has Beneficiary
     Expected: False
     Actual:   None
```

Cause: Validator couldn't find the boolean value

Debug Steps:

1. Open CSV and find `hasBeneficiary` row
2. Check if it has `click` action
3. Verify XPath contains `label[1]` or `label[2]`

Solution: The `get_boolean_value()` function detects booleans from XPath patterns.

Issue 3: API Key Error

```
openai.AuthenticationError: Invalid API key
```

Solution:

1. Check `.env` file exists in parent directory
2. Verify key format: `OPENAI_API_KEY=sk-...`
3. Ensure no extra spaces or quotes

Issue 4: Context Too Large

```
openai.BadRequestError: maximum context length exceeded
```

Cause: Schema + rules + example exceeds model limit

Solution:

- GPT-4o supports 128K tokens (should not hit this)
- If using gpt-4o-mini, may need to truncate example CSV

Issue 5: Wrong XPaths in Output

```
Element not found: //*[@id="nonexistent"]
```

Cause: LLM invented an XPath instead of using schema

Solution:

1. Check if element exists in `form_schema.json`
2. Add missing element to schema
3. Re-run generation

Debug Mode

Add verbose logging:

```python
# In generate_test_case.py, add before API call:
print(f"System prompt length: {len(system_prompt)} chars")
print(f"Estimated tokens: {estimate_tokens(system_prompt)}")
```

---

15. Limitations & Known Issues

Current Limitations

Limitation	Impact	Workaround
XPath changes break tests	Form updates require schema update	Maintain schema as living document
~16K token context	Can't add heavy data	Consider RAG for very large forms
$0.06/call cost	Adds up for thousands of tests	Use gpt-4o-mini for simple scenarios
Boolean detection heuristic	May fail for non-standard forms	Update `get_boolean_value()` logic
English-only scenarios	Can't parse other languages	Add multilingual parsing

Known Issues

1. **Inconsistent OTP count:** Sometimes generates 4 OTP entries instead of 2 (passes validation with `>=2` check)

2. **Beneficiary percentage rounding:** For 3 beneficiaries, may generate 33%, 33%, 33% = 99% instead of 34%

3. **Complex conditionals:** Deeply nested conditions (if X and Y and Z) may not always apply correctly

4. **File upload paths:** Uses placeholder paths that need to be replaced with actual files

---

16. Future Enhancements

Short-Term (Next Sprint)

Enhancement	Benefit
Batch generation from matrix	Generate 100 scenarios from combinations
Selenium/Playwright integration	Direct execution of generated CSVs
Web UI	Non-technical users can generate tests
Stricter validation	Exact OTP count, percentage sum checks

Medium-Term

Enhancement	Benefit
RAG integration	Vector DB for large schemas
CI/CD pipeline	Auto-generate tests on PR
Test execution reporting	Track pass/fail rates
Schema diffing	Detect form changes automatically

Long-Term (Future)

Enhancement	Benefit
Multi-form support	Single system for multiple applications
Visual form analysis	LLM reads screenshot to update schema
Self-healing XPaths	Auto-detect and fix broken locators
Natural language test results	"Test passed but took 30s longer than usual"

Appendix A: Full System Prompt

The system prompt sent to GPT-4o includes:

1. **Role definition** - "You are a test case generator..."
2. **Form structure overview** - 6 pages, their purposes
3. **Output format specification** - CSV columns, quoting rules
4. **Critical rules** - Sequence, dropdowns, booleans, conditionals
5. **Repeatable section rules** - Beneficiaries, joint partners, dependents
6. **Complete conditional_rules.json** - All 12 rules

7. **Complete form_schema.json** - All pages, fields, XPaths
8. **Complete example CSV** - 154-row reference
9. **Generation instructions** - Think through employment, booleans, repeatables

Total: ~16,000 tokens

Appendix B: CSV Format Reference

```
,Group,Element,Action,Value,Strategy,XPath
0,Contact Info,firstName,click,,id,"//*[@id=""firstName""]"
1,Contact Info,firstName,Input,JOHN,id,"//*[@id=""firstName""]"
2,Contact Info,email,click,,id,"//*[@id=""email""]"
3,Contact Info,email,Input,test@example.com,id,"//*[@id=""email""]"
```

Column	Description	Examples
(index)	Row number, 0-based	0, 1, 2, ...
Group	Page/section name	"Contact Info", "Documents"
Element	Field ID or button text	"firstName", "Save & Continue"
Action	Operation type	"click", "Input"
Value	Data to enter	"JOHN", "test@example.com", ""
Strategy	Locator method	"id", "data-testid", "absolute"
XPath	Element locator	"//*[@id="firstName"]"

Appendix C: Glossary

Term	Definition
LLM	Large Language Model (e.g., GPT-4o)
Token	Unit of text for LLM (~4 characters)
XPath	XML Path expression to locate HTML elements
Schema	Structured definition of form elements
Conditional Rule	If-then logic for field visibility
Few-shot Learning	Teaching LLM by example
PEP	Politically Exposed Person
FATCA	Foreign Account Tax Compliance Act
LinCU	Credit union debit card product
FIP	Financial Insurance Plan

Term	Definition
OTP	One-Time Password

End of Manual