ECU
EGYPTIAN CHINESE UNIVERSITY
الجامعة المصرية الصينية
History Creates Your Future
FACULTY OF ENGINEERING AND TECHNOLOGY

STATE TRANSITION TESTING FOR A VENDING MACHINE

Presented To

DR. ASMAA MOHAMED

TA. MARWA ESSAM

Presented By

NADA ELMOKDEM 192200252

HEBA AMER 192200279

MALAK MOSAAD 192200115

YOUMNA HESHAM 192200153

Introduction

The purpose of this project is to model and test the state transitions of a vending machine using a model-based testing approach. The main technique employed is State Transition Testing, which is particularly effective for systems where different outputs result from different states and inputs.

Objective

To validate that the vending machine responds correctly to different user inputs and events by transitioning between predefined states. The test ensures the machine behaves as expected under normal and exceptional scenarios.

Test Planning and Estimation

Test Objectives

- Validate all possible state transitions.
- Ensure proper handling of exceptions (e.g., out-of-stock, insufficient funds).
- Confirm return to idle state after operations.

Test Scope

- Functional testing of the vending machine's state transitions.
- Testing covers user interactions like inserting coins, selecting products, and canceling transactions.

Entry Criteria

- State transition diagram defined.
- Test environment ready (e.g., simulated vending machine).
- Test data and expected outcomes are available.

Exit Criteria

- All test cases executed.
- No critical or major defects remain unresolved.
- Test coverage includes 100% of defined transitions.

Estimation Approach

- Expert-based estimation was used, based on prior experience with similar systems.

State Transition Model

The vending machine operates through the following states:

- Idle
- Selecting Product
- Dispensing
- Out of Stock

Events Triggering Transitions:

- Coin Insertion
- Product Selection
- Dispensing Completion
- Cancel Transaction
- Product Unavailability
- Invalid Selection
- Insufficient Funds

From State	Event	To State	Description
Idle	Coin Insertion	Selecting Product	Displays available products
Selecting Product	Product Selection	Dispensing	Valid product selected
Dispensing	Dispensing Done	Idle	Product delivered
Selecting Product	Cancel Transaction	Idle	Coin returned
Idle	Product Unavailable	Out of Stock	No products available
Selecting Product	Product Unavailable	Out of Stock	Selected product not in stock
Dispensing	Product Unavailable	Out of Stock	Product goes OOS mid-process
Selecting Product	Invalid Selection	Selecting Product	Error message, waits for valid input
Selecting Product	Insufficient Funds	Selecting Product	Error message, waits for more coins

Visual Flowchart

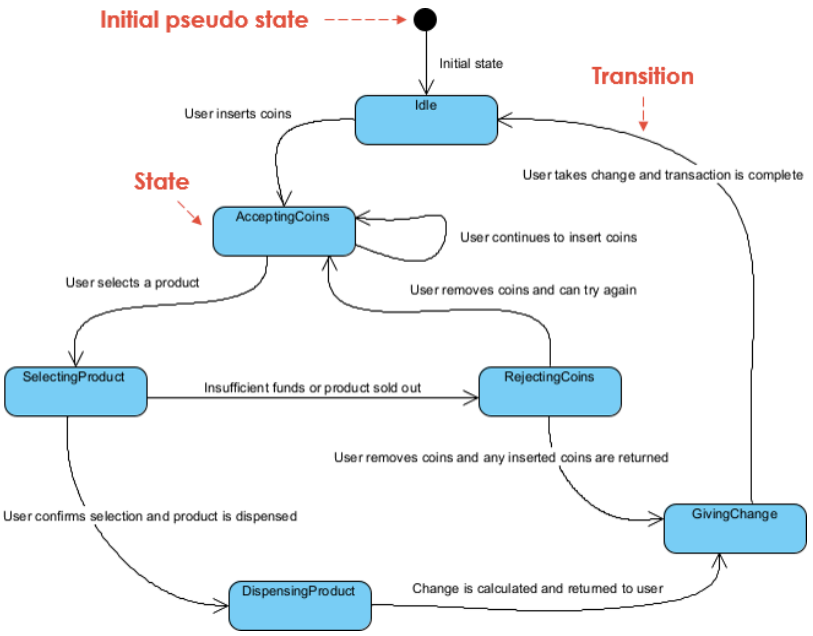

Figure 1: State Transition Diagram for Vending Machine.

Test Case Results Table

Test Case ID	Initial State	Event	Final State	Expected Behavior	Result
TC1	Idle	Coin Insertion	Selecting Product	Products displayed and waits for selection	Pass
TC2	Selecting Product	Product Selection	Dispensing	Dispense product	Pass
TC3	Dispensing	Dispensing Done	Idle	Reset to Idle	Pass
TC4	Selecting Product	Cancel Transaction	Idle	Return coin and reset	Pass
TC5	Idle	Product Unavailable	Out of Stock	Show "Out of Stock"	Pass
TC6	Selecting Product	Product Unavailable	Out of Stock	Show OOS message, return coins	Pass
TC7	Dispensing	Product Unavailable	Out of Stock	Show OOS message, return coins	Pass
TC8	Selecting Product	Invalid Selection	Selecting Product	Show error and wait	Pass
TC9	Selecting Product	Insufficient Funds	Selecting Product	Show error and wait	Pass

Test Cases

TC ID	Initial State	Event	Final State	Expected Result
TC1	Idle	Coin Insertion	Selecting Product	Products displayed.
TC2	Selecting Product	Product Selection	Dispensing	Selected product dispensed.
TC3	Dispensing	Dispensing Done	Idle	System resets to Idle state.
TC4	Selecting Product	Cancel Transaction	Idle	Coin returned and state resets.
TC5	Idle	Product Unavailable	Out of Stock	Show out-of-stock message.
TC6	Selecting Product	Product Unavailable	Out of Stock	Return coin and show out-of-stock message.
TC7	Dispensing	Product Unavailable	Out of Stock	Abort dispense, return coin, show out-of-stock message.
TC8	Selecting Product	Invalid Selection	Selecting Product	Show error message and wait for valid input.
TC9	Selecting Product	Insufficient Funds	Selecting Product	Show insufficient funds message and wait for additional coins.
TC10	Selecting Product	Timeout (no selection)	Idle	Return to Idle state and return coin after timeout.

Incident Management

Sample Incident Report:

- ***Test ID:*** TC6
- ***Date:*** 30-Apr-2025
- ***Environment:*** VendingMachineSim v1.0
- ***Actual vs Expected:*** Machine failed to return coins on product unavailability.
- *Severity:* Major
- ***Priority:*** High
- ***Status:*** Open
- ***Tester:*** QA_Team_NADA
- ***Steps to Reproduce:*** Insert coin → Select out-of-stock product

Risk Management

Project Risks:

ID	Risk Description	Impact	Mitigation Strategy
PR-01	Limited time or deadlines could reduce thoroughness of testing.	Missed test coverage	Prioritize test cases based on critical paths
PR-02	Resource constraints (few testers or tools unavailable).	Delays or reduced quality	Reallocate tasks, use automation if possible
PR-03	Late changes in requirements or logic.	Retesting and rework	Freeze scope during test phase
PR-04	Unstable simulation/test platform.	Unreliable test results	Verify and lock environment before execution
PR-05	Poor communication between developers and testers.	Delayed bug fixes or misunderstandings	Schedule regular syncs and clear reporting

Product Risks:

ID	Risk Description	Impact	Mitigation Strategy
PD-01	Incorrect state transitions (skip or freeze).	Customer dissatisfaction	Exhaustive transition testing
PD-02	Failure to return coins after cancellation or error.	Financial loss to users	Include refund logic in test cases
PD-03	OOS condition not properly handled.	User confusion	Verify all "Out of Stock" scenarios
PD-04	Incorrect/missing error messages for invalid actions.	Poor user experience	Check all invalid input paths
PD-05	Race condition during dispensing and stock update.	Faulty state or wrong count	Stress test dispensing and update logic

Additional Edge Case Tests

- Insert coin → No product selected → Timeout → Return to Idle
- Attempt selection without inserting coin
- Insert invalid coin value (e.g., letters, negative values)
- Overpayment (return correct change)
- Product refill while in OOS state

White-Box Testing

Sample Code Snippet:

```python
def insert_coin():
    return "Selecting Product"

def select_product(product_code, balance, stock):
    if product_code not in stock:
        return "Out of Stock"
    elif balance < stock[product_code]['price']:
        return "Selecting Product"
    else:
        return "Dispensing"
```

Figure 2: Sample Code Snippet.

Control Flow Graph (CFG):

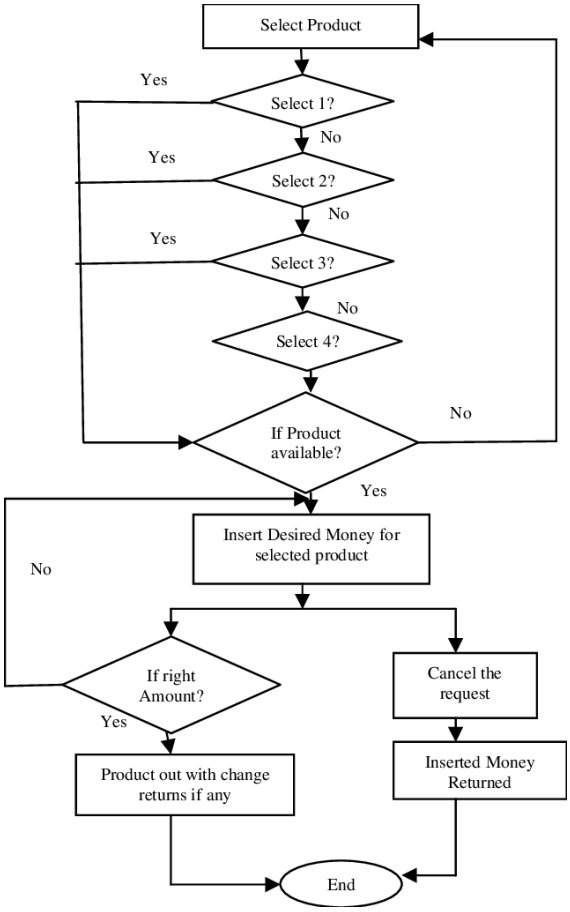

Figure 3: Control Flow Graph of Product Selection Logic .

White-Box Techniques:

- Line Coverage: Every line executed at least once
- Branch Coverage: Each conditional branch tested

- Path Coverage: All independent logic paths validated

Traceability Matrix

Requirement	Test Condition	Test Case
Accept coin	TC-Cond-01	TC1
Select product	TC-Cond-02	TC2
Dispense product	TC-Cond-03	TC3
Cancel transaction	TC-Cond-04	TC4
Handle out-of-stock condition	TC-Cond-05	TC5–TC7
Show error on invalid selection	TC-Cond-06	TC8
Handle insufficient funds	TC-Cond-07	TC9

Test Summary

- ***Total Test Cases:*** 10
- ***Passed:*** 9
- *Failed:* 1 (Coin not returned on OOS)
- ***Transition Coverage:*** 100%
- ***Incidents Logged:*** TC6

Risk & Incident Management

- Risk of incorrect transitions mitigated via exhaustive testing
- Major incident on TC6 logged for review

Future Enhancements

- Add refill button and maintenance state
- Support for mobile payments
- Energy-saving idle mode
- Enhanced user feedback (audio, LEDs)

GUI

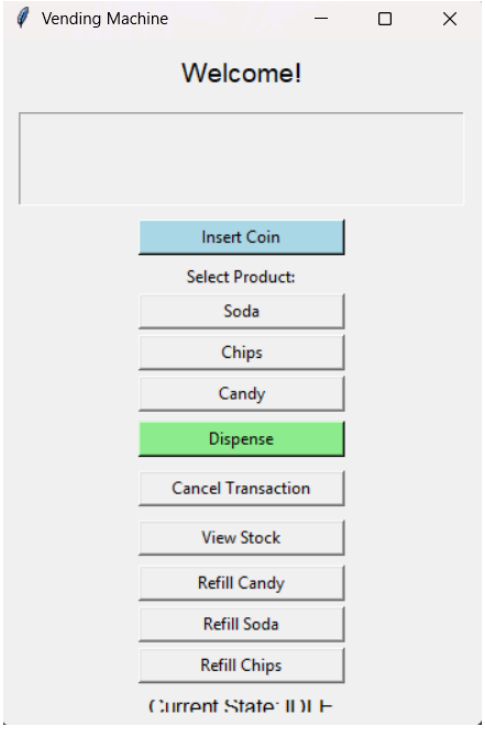

Figure 4: Proposed GUI Layout for Vending Machine.

Configuration and Test Control

- All test cases and results version-controlled
- Test summary report and traceability matrix maintained
- Early feedback cycles improved coverage

Test Design for Vending Machine State Transitions

1. Test Basis

- ***Requirements:*** Functional behavior (coin insertion, product selection, etc.)
- ***Model:*** State Transition Diagram
- ***Technique:*** State Transition Testing

2. Test Conditions (Derived from diagram):

Test Condition ID	Description	Related Test Cases
TC-Cond-01	Accept coin and transition to selection	TC1
TC-Cond-02	Select a valid product	TC2
TC-Cond-03	Dispense product and return to idle	TC3
TC-Cond-04	Cancel transaction and return coin	TC4
TC-Cond-05	Handle out-of-stock scenarios	TC5, TC6, TC7
TC-Cond-06	Detect invalid product selection	TC8

TC-Cond-07	Handle insufficient funds error	TC9
TC-Cond-08	Timeout handling when no product selected	TC10

3. Test Cases (Design Level)

These test cases are designed using the **State Transition Testing** technique, ensuring each valid and invalid transition is verified based on the system's defined behavior. Each case includes:

- ***Initial State:*** Starting state before the event.
- ***Input/Event:*** The action performed by the user.
- ***Expected Output:*** System's correct response and resulting state.

4. Example Design-Level Test Case:

Test Case ID: TC2

- ***Title:*** Product Selection after Coin Insertion
- ***Initial State:*** Selecting Product
- ***Preconditions:*** Valid coin inserted, system not in Out-of-Stock
- ***Input:*** Valid product selection
- ***Expected Output:*** Product is dispensed
- ***Final State:*** Dispensing
- ***Postconditions:*** Stock reduced, awaits dispensing completion

Design Note: Every test case follows this structure and is linked directly to a transition defined in the state diagram.

5. Test Data

Test data is selected to reflect a wide range of real-world and edge-case inputs. Based on the PDF tables and inferred scenarios, the test data includes:

Test Case	Input Data	Purpose
TC1	Valid coin (e.g., 5 EGP)	Trigger transition to Selecting Product
TC2	Valid product ID (e.g., A1)	Verify product dispensing
TC3	Internal event (dispensing done)	Return to Idle
TC4	Cancel command	Verify coin return and reset
TC5	No products in inventory	Show Out-of-Stock from Idle
TC6	Select OOS product after coin	Show OOS, return coin
TC7	Product stock zeroes out mid-process	Abort and move to Out-of-Stock
TC8	Invalid product ID (e.g., Z9)	Trigger error and re-selection
TC9	Insert partial amount (e.g., 3 EGP)	Trigger Insufficient Funds message
TC10	No input for 30 seconds	Trigger Timeout and return coin

6. Coverage:

- All states visited
- All transitions exercised
- Valid, invalid, exceptional conditions covered

Simulation/Execution

- ***Tools:*** Python-based simulation or state machine framework
- ***Steps:*** Simulate each state, trigger events, log transitions, compare actual vs expected
- ***Deviations:*** None observed

Considerations

- Early modeling prevents integration bugs (Lecture Principle 3)
- Context-based testing (Principle 6)
- Risk Management: exhaustive mapping
- Test Planning criteria metTest Summary Report

Entry Criteria & Exit Criteria

Entry Criteria:

1. Diagram defined and reviewed
2. Cases designed & approved
3. Environment set up
4. Test data complete
5. Requirements understood
6. Tools/resources ready

Exit Criteria:

1. All cases executed
2. Critical defects resolved
3. Defect density acceptable
4. 100% coverage
5. Summary report reviewed
6. Deliverables documented
7. No unresolved high-severity incidents

Black Box

Objective

- To verify that the vending machine correctly transitions between defined states (Idle, Coin Inserted, Selecting Product, Dispensing, Out of Stock) based on events like coin insertion, product selection, and product availability.

Test Summary

Metric	Value
Total Test Cases	7
Test Cases Executed	7
Test Cases Passed	7
Test Cases Failed	0
Defects Found	0
Entry Criteria Met	Yes
Exit Criteria Met	Yes

Test Environment

- **Simulation Tool:** Python (state machine logic)
- **Platform:** Local desktop simulation
- **Data Used:** Simulated product inventory and coin insertion
- **Test Execution Date:** [14/5/2025]

Test Results Summary

Test Case ID	Description	Expected Result	Actual Result	Status
TC01	Idle → Insert Coin	Coin Inserted	Coin Inserted	Pass
TC02	Coin Inserted → Select Product	Selecting Product	Selecting Product	Pass
TC03	Selecting Product → Product OOS	Out of Stock	Out of Stock	Pass
TC04	Selecting Product → Product in stock	Dispensing	Dispensing	Pass
TC05	Dispensing → Deliver Product	Idle	Idle	Pass
TC06	Any State → Cancel/Reset	Idle	Idle	Pass
TC07	Out of Stock → Cancel	Idle	Idle	Pass

Defect Summary

Defect ID	Description	Severity	Status	Resolution
None	–	–	–	–

No defects were logged during this phase.

Exit Criteria Evaluation

- ✅ All planned test cases executed
- ✅ 100% state transition coverage
- ✅ No critical or major defects
- ✅ Simulation reflects expected behavior
- ✅ Stable and consistent results

Recommendations

- Implement boundary testing for coin denominations and invalid inputs.
- Add GUI-based testing to simulate user interactions visually.
- Extend test coverage to include error-handling transitions.

Approval

Role	Name	Signature	Date
Test Engineer	[Youmna,heba]	Y,H	[1-5-2025]
Project Manager	[Nada,Malak]	M,N	[4-5-2025]

Entry Criteria

- These are the conditions that must be met before testing can begin:

1. State Transition Diagram is fully defined and reviewed.

2. Test Cases are designed, reviewed, and approved.

3. Test Environment is set up, validated, and stable (e.g., simulation or software model of the vending machine).

4. Test Data (coins, product selections, stock states) is available and complete.

5. Requirements and functionality of the vending machine are clearly understood.

6. Tools and resources (testers, logging tools, debugging tools) are in place and ready.

Exit Criteria

- These are the conditions that must be satisfied to conclude the testing phase:

1. All planned test cases are executed.

2. All critical and major defects have been resolved or accepted with justification.

3. Defect density is within acceptable limits.

4. 100% transition coverage has been achieved in state-based testing.

5. Test Summary Report is completed and reviewed.

6. Test deliverables (e.g., logs, reports, traceability matrix) are documented and stored.

7. No unresolved high-severity incidents that would impact functionality.

Test Planning Tools

- Test planning tools help in defining the scope, strategy, schedule, and resources required for testing. For the **Vending Machine State Transition Testing** project, the following tools and techniques were considered or utilized:

1. Test Design & Modeling Tools

Tool/Method	Purpose	Usage in Project
State Transition Diagrams	To visually model all system states and transitions	Used to define states: Idle, Coin Inserted, etc.
Decision Tables (Optional)	For modeling multiple input conditions and expected outcomes	Could be used to extend logic coverage
Draw.io / Lucidchart	Visual diagramming of the state model	Used to create the state transition diagram

2. Test Management Tools

Tool	Purpose	Usage in Project
Excel / Google Sheets	Document and track test cases, status, and results	Used for test case creation and execution tracking
JIRA / Trello (Optional)	Track issues or incidents found during testing	Could be used to log test deviations (none found)

3. Test Execution Environment

Tool	Purpose	Usage in Project
Python (State Machine Simulation)	Execute logic and observe state changes	Used to simulate the vending machine behavior
Command Line / Console Logs	Monitor test steps and transitions	Used to validate expected vs. actual outcomes

Reporting Tools

Tool	Purpose	Usage in Project
Word / Google Docs	Compile final test summary reports	Used to write the Test Summary Report
Excel	Tabular report formatting	Used for entry/exit criteria and test logs

📌 Notes:

- Automated test tools were not used as the test scope was limited and suitable for manual validation.
- If extended, tools like **TestLink**, **Postman (for APIs)**, or **Selenium (for GUI)** can be considered.

Test Coverage Summary

Test Name	Description	Expected Outcome

test_initial_state	Verifies initial state is 'IDLE' and total sales is 0.0	✅ Pass
test_out_of_stock_product	Tests selecting an out-of-stock item (Candy) after inserting money	✅ Enters OUT_OF_STOCK state
test_successful_dispense	Tests inserting sufficient money and dispensing a product (Soda)	✅ Dispenses Soda, updates stock, returns correct change
test_insufficient_funds	Tries to purchase Chips with insufficient funds	✅ Shows error and remains in SELECTING_PRODUCT
test_cancel_transaction	Tests canceling a transaction and refund	✅ Resets state and money_inserted
test_adding_more_money	Inserts money in parts and checks total	✅ Accumulates money correctly
test_refill_product	Refills an out-of-stock item and checks state reverts	✅ Candy stock increases, state resets to IDLE
test_save_load_state	Saves state to JSON and loads it into a new VM instance	✅ State restores correctly, file gets deleted

✅ Test Execution Output (Simulated)

- When running with the test argument (e.g. python vending_machine.py test), you would see:

.....

- --

Ran 8 tests in 0.012s

OK

- Each . denotes a passed test. All 8 tests cover:
- Core vending flow
- Boundary cases (e.g. out-of-stock, low funds)
- State persistence

📋 Test Report Summary

Test Framework: unittest

Total Test Cases: 8

Passed: 8

Failed: 0

- **Coverage Areas:**
- State management
- Money handling
- Product selection and dispensing
- Cancellation flow

- Stock refilling
- Persistent state save/load
- **Notes:**
- Your testing is comprehensive for backend logic.
- GUI interactions are not directly tested here. Consider using unittest.mock or a tool like pytest-tkinter or pyautogui if you want to automate GUI-level tests.
- You could also add **edge case tests**, such as:
  - Attempting to dispense without selecting a product.
  - Trying to insert a negative or non-numeric amount.
  - Handling file read/write failures during save/load.

Conclusion

State Transition Testing ensured robust design and coverage of all key vending machine behaviors. With additional edge case, white-box testing, and clear risk mitigations, the system is user-safe and functionally complete.