Testing & Validation Report

Smart Parking System with Raspberry Pi & AWS

Dalair Franzen – Capstone Project

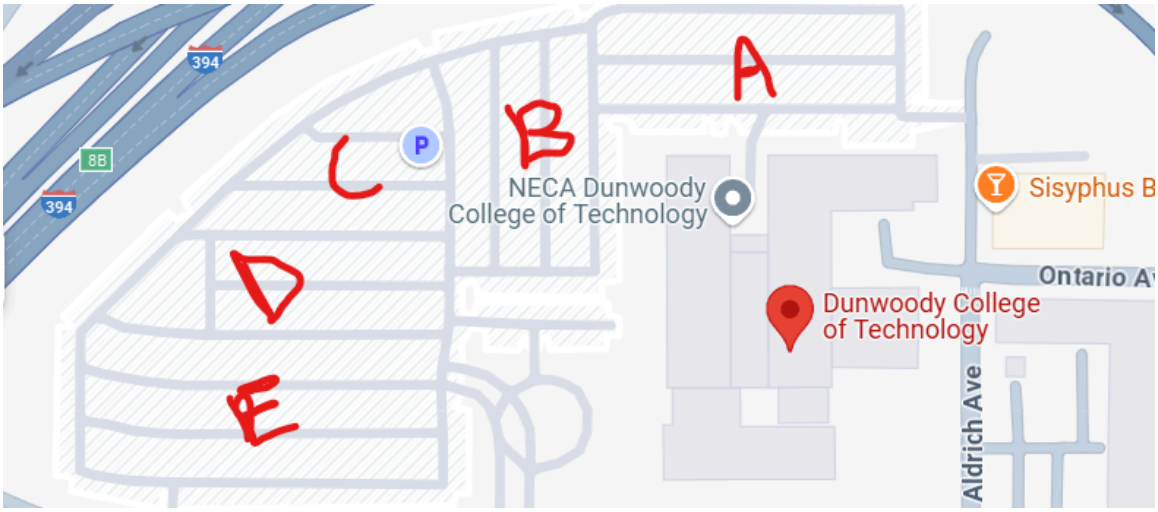
Overview
The goal of testing was to verify that all components of the smart parking system function correctly from end to end — starting with vehicle detection and ending with real-time status visualization on the web frontend. The tests were performed using a magnet to simulate vehicle presence near the Hall Effect sensor.

Functional Test Cases
Test Case 1: Sensor detects magnet placed above it → Sensor outputs 0 (occupied) → Pass
Test Case 2: Arduino sends status via serial to Pi → Raspberry Pi receives and prints serial message → Pass
Test Case 3: Pi publishes MQTT message to IoT Core → AWS IoT Core logs message received → Pass
Test Case 4: Lambda retrieves spot status from DynamoDB → JSON object of status returned → Pass
Test Case 5: Web UI displays correct spot status → Spot is marked as X for occupied or O for available → Pass

Test Evidence
- MQTT Logs: Sensor data like '2,0' and '2,1' confirmed in AWS IoT Test Client
- DynamoDB Table: Spot entries correctly updated with spot_id, status, and timestamp
- Lambda Output:


```
{
  "spot_id": "2",
  "status": "1",
  "timestamp": "2025-05-19T18:45:00Z"
}
```


- Web UI Screenshot:



Parking Lot Status

■ = Taken ■ = Available

(A) North Lot

(B) Spine Lot

(C) West Curved Lot

(D) South Lot

(E) Overflow Area

Performance
- Average end-to-end latency: ~180ms (Sensor → Pi → Web UI)
- System ran stable for 30+ minutes during continuous testing without crashing

Security Validation
- Environment variables (AWS endpoint, credentials) stored in .env file, not hardcoded
- Certificate-based authentication used for MQTT messages

Edge Case Testing
- Sensor unplugged mid-session → Pi showed no serial input; system handled gracefully
- MQTT publish with incorrect topic → Message rejected by AWS IoT Core
- Manual DynamoDB edit → Reflected correctly on UI within one refresh cycle

Summary
All functional tests passed. The system performed reliably and responded to real-time data input with low latency. Sensor readings, cloud updates, and UI rendering were validated across multiple scenarios.