SHOPPING MALL PARKING MANAGEMENT SYSTEM

A TERMINAL-BASED PARKING SIMULATOR INSPIRED BY MALAYSIA'S 1 UTAMA

INSPIRATION

Modeled after 1 Utama's real parking system. Each zone, membership tier, and day type affects fee logic. Built entirely in Python to run in a text-based terminal.

CORE FEATURES

Zones: Regular, Preferred, Valet, Outdoor, Staff
Memberships: Non-Member, Member, Silver, Gold, Staff
Policies:

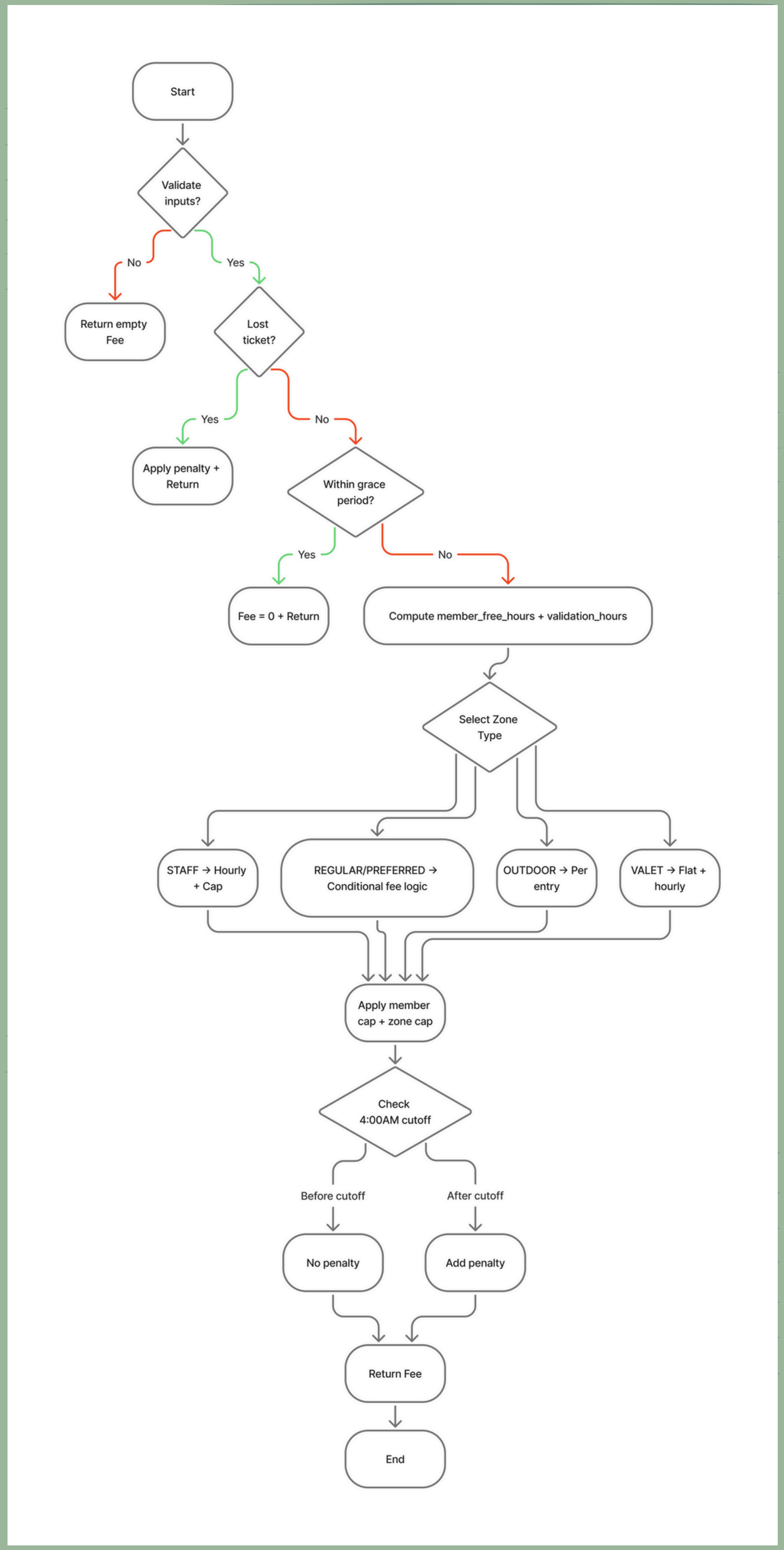
- Grace period (15 min free)
- Variable rates per zone & day type
- 4:00 AM overnight penalty
- Lost-ticket flat penalty
- Daily caps
- Validation bonuses → valid stores with acquired minimum spent → 2 free hours parking

PROJECT STRUCTURE

```
src/
 ├ fee_engine.py     → core calculation logic
 ├ policy.py         → pricing, caps & validation rules
 ├ data_manager.py → JSON loader
 └ ui.py             → all terminal logic
main.py             → entry point
tests/              → all test suites
```

Built and Tested by
Rachel Tham Wing Yern
34896813

CONTROL FLOW OF COMPUTE_FEE



TEST-DRIVEN WORKFLOW

- fee_engine → black-box first
- White-box & control flo analysis after implementation
- ui.py → mock & input validation
- ApprovalTests last for receipt verification

BLACK-BOX TESTING

Key Equivalence Partitions:

- Grace: 14 min vs 15 min
- Cutoff: 3:59 AM / 4:00 AM / 4:01 AM
- Day type: Weekday vs Weekend vs Public Holiday
- Membership tier: Non-member vs Member vs Gold
- Zone: Regular vs Preferred vs Valet vs Outdoor vs Staff
- Validation: Spend < 30 vs ≥ 30
- Lost-ticket: True / False

WHITE-BOX TESTING

Executed paths for:

- Missing policy → return 0
- Lost-ticket → penalty branch
- Grace period → immediate return
- Every zone path (5 distinct calculations)
- Cap application (member + zone)
- Cutoff exception branch (invalid timestamps)

COVERAGE RESULTS

- Line Coverage = 82%
- Branch Coverage = 82%

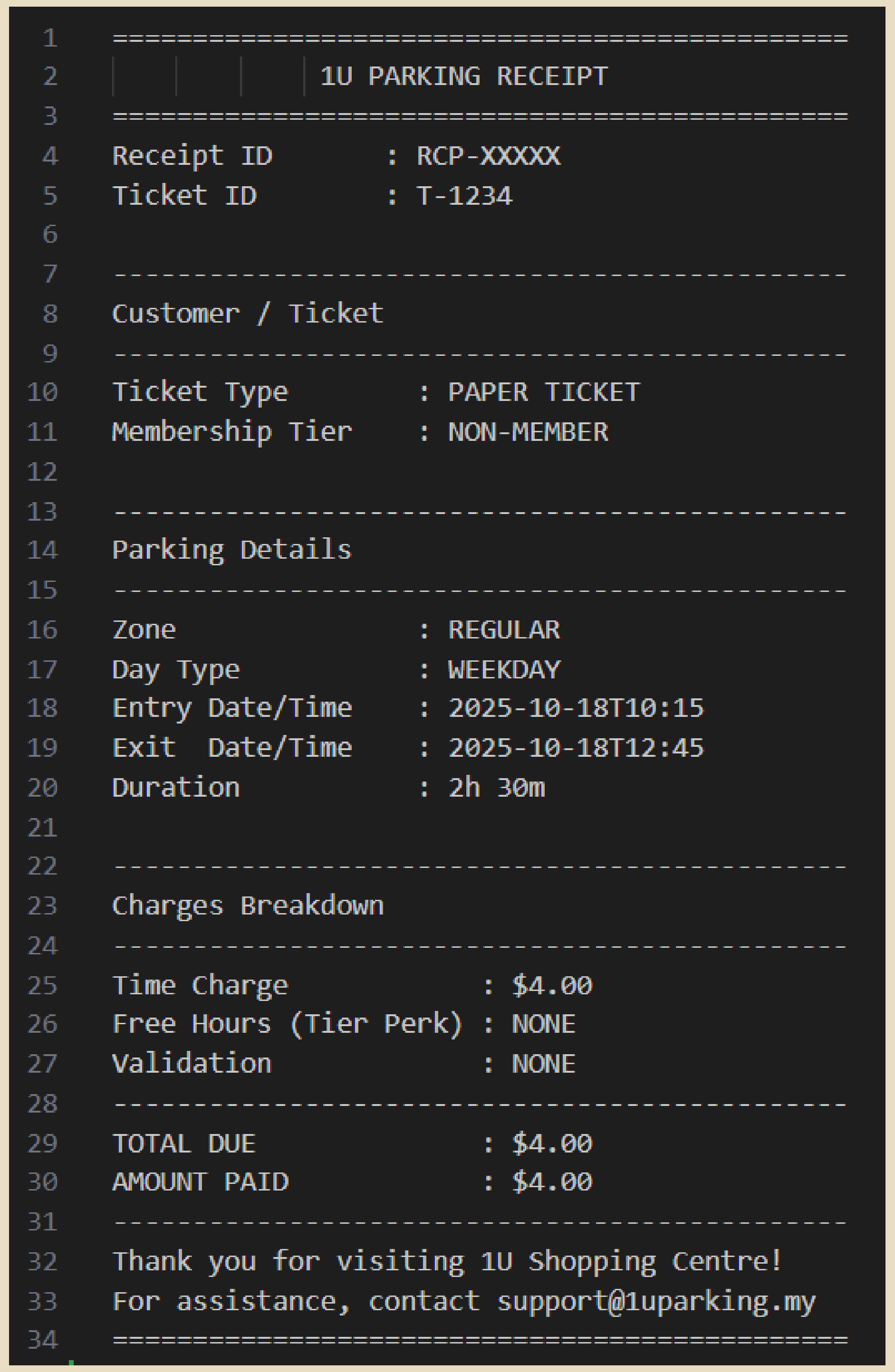
MOCK TESTING

Tested terminal UI without real input or files by patching:

- builtins.input → simulated menu choices
- load_tickets() → fake JSON records
- compute_fee() → injected Fee stubs

APPROVALTESTS

Compared full printed receipts line-by-line with approved snapshots, with an example test case below
Non-member weekday (2 h 30 m):



```
==============================================
            1U PARKING RECEIPT
==============================================
Receipt ID        : RCP-XXXXX
Ticket ID         : T-1234


----------------------------------------------
Customer / Ticket
----------------------------------------------
Ticket Type          : PAPER TICKET
Membership Tier      : NON-MEMBER


----------------------------------------------
Parking Details
----------------------------------------------
Zone                 : REGULAR
Day Type             : WEEKDAY
Entry Date/Time      : 2025-10-18T10:15
Exit  Date/Time      : 2025-10-18T12:45
Duration             : 2h 30m


----------------------------------------------
Charges Breakdown
----------------------------------------------
Time Charge                : $4.00
Free Hours (Tier Perk)     : NONE
Validation                 : NONE
----------------------------------------------
TOTAL DUE                  : $4.00
AMOUNT PAID                : $4.00
----------------------------------------------
Thank you for visiting 1U Shopping Centre!
For assistance, contact support@1uparking.my
==============================================
```

If spacing, labels, or totals differ → test fails. This ensures consistent formatting, layout, and data presentation.

SHORT REFLECTION

Through this project, I learned that testing is a design discipline, the way it doesn't just verify correctness; it builds confidence, consistency, and clarity in how software behaves.