

System Design: Component Implementation

Name: Shrey Deshmukh

Student ID: 82459391

Net ID: sdeshmu2

MCS 253P- Advanced Programming and Problem Solving

Introduction

This project implements the backend logic for the **Hampr Machine Service**, a simulated laundry machine management system.

API Handler Code Summary

The ApiHandler class is the core component responsible for routing and executing requests. It integrates:

- **IdentityProviderClient** - validates tokens
- **MachineStateTable** - simulates persistent machine state
- **SmartMachineClient** - simulates hardware API calls
- **DataCache** - reduces DB load by caching machine states

Key Features Implemented

1) Token Validation

checkToken() calls the IDP client, and if invalid, throws a structured JSON error

2) Machine Reservation

- Fetches all machines at the requested location
- Selects the first AVAILABLE machine
- Updates its status to AWAITING_DROPOFF
- Assigns the user's job ID
- Caches the updated machine

3) Machine Lookup

- Returns data from cache when available
- Falls back to DB otherwise
- Stores DB results back into cache

4) Starting a Machine

- Only machines in AWAITING_DROPOFF can be started
- Calls external Smart Machine API:
 - On success - machine becomes RUNNING
 - On failure - machine becomes ERROR
- Updated state is committed to DB and cache

Analysis

1) API Behavior

- The handler correctly separates responsibilities: token validation, routing logic, business logic, cache lookups, and database persistence.
- Caching strongly reduces redundant database reads, demonstrated by a stable ~65% hit rate.
- The error-handling paths (invalid token, unavailable machine, hardware failure) behave exactly as tests expect.
- The SmartMachine hardware failure logic correctly transitions machines to ERROR, ensuring a consistent system state.

2) System Characteristics

- The caching layer is highly beneficial, improving throughput and lowering DB load.
- The architecture follows clean boundaries between components, making it extendable for a real backend.
- Simulation results confirm that the system behaves predictably under repeated runs.

Attached Performance Screenshots

```
C:\UCI\253P-APPS\System_Design_Github_Project\cs-253p-hw-hampr-machine-service>npm test

> hampr-base@1.0.0 test
> jest

PASS test/api.test.ts
PASS test/simulation.test.ts
  • Console

  console.log
    (index) | Resource | Run 1 Units | Run 1 % | Run 2 Units | Run 2 % | Run 3 Units | Run 3 % | Run 4 Units |
    Run 4 % |
    0 | 'IdentityProviderClient' | 3840256 | '69.54%' | 3840256 | '69.16%' | 3840256 | '69.38%' | 3840256 |
    '69.11%' |
    1 | 'SmartMachineClient' | 32256 | '0.58%' | 32256 | '0.58%' | 32256 | '0.58%' | 32256 |
    '0.58%' |
    2 | 'MachineStateTable' | 1661128 | '30.08%' | 1661128 | '29.91%' | 1661128 | '30.01%' | 1661128 |
    '29.89%' |
    3 | 'DataCache' | 23002 | '0.42%' | 23002 | '0.41%' | 23002 | '0.42%' | 23002 |
    '0.41%' |

    at Object.<anonymous> (test/simulation.test.ts:160:13)

  console.log
```

(index)	Run	Cache Hits	Cache Misses	Hit Rate
0	1	3949	2115	'65.12%'
1	2	3831	2151	'64.04%'
2	3	3901	2115	'64.84%'
3	4	3816	2160	'63.86%'

```
    at Object.<anonymous> (test/simulation.test.ts:161:13)

  console.log
    (index) | Run | Cache Hits | DB Accesses | Hit/Access Ratio |
    0 | 1 | 3949 | 6584 | '0.5998' |
    1 | 2 | 3831 | 6584 | '0.5819' |
    2 | 3 | 3901 | 6584 | '0.5925' |
    3 | 4 | 3816 | 6584 | '0.5796' |

    at Object.<anonymous> (test/simulation.test.ts:162:13)

Test Suites: 2 passed, 2 total
Tests: 12 passed, 12 total
Snapshots: 0 total
Time: 3.492 s, estimated 4 s
Ran all test suites.

C:\UCI\253P-APPS\System_Design_Github_Project\cs-253p-hw-hampr-machine-service>
```

