UNIVERSITI TEKNOLOGI PETRONAS

TEB2014 SOFTWARE ENGINEERING AND HCI

SEPT 2025 SEMESTER

GROUP PROJECT:

FUEL SUBSIDY MANAGEMENT SYSTEM

No.	Name	Student ID
1	Low Ren Jing	22011246
2	Noor Zaheera Binti Noor Zahidi	22011059
3	Ammirrul Adam Bin Amran	22010256
4	Muhammad Aqil Rahimi Bin Mohamad Rasidi	22011363
5	Putri Humaira Kaur Binti Hamidon	22011092

TABLE OF CONTENTS

SECTION 1: SOFTWARE ENGINEERING

1.1 REQUIREMENTS SPECIFICATIONS

1.1.1 Functional and Non-Functional Requirements

Functional Requirements

MUST

1. **Real–time Eligibility API**
   - POS can check with Citizen ID, license, card, vehicle info, requested litres or amount. System returns eligibility status, subsidy amount, remaining quota, and a transaction authorization token.
2. **Citizen Registration and Verification**
   - The system must allow citizens to register using their MyKad and Driver's License. The system must verify that the user is a valid registered driver through integration with national providers, ensuring that underage individuals cannot create an account or claim subsidies.
3. **Transaction Authorization and Recording**
   - The system must authorize and record every subsidised fuel transaction using an authorization token, ensuring accurate tracking of subsidy usage.
4. **Quota Management**
   - The system must maintain and update each citizen's remaining subsidy quota to prevent overuse.

SHOULD

1. **Integration with Payment Gateway**
   - The system should integrate with payment providers to automatically handle co-payments.

COULD

1. **Citizen Notification**
   - Citizens could receive SMS or app notifications after each subsidised purchase
2. **Multi – Language Support**
   - The system could support Malay and English for wider accessibility.

WON'T

1. **Digital Wallet Management**
   - The system will not handle citizens' personal e- wallets, it only integrates with existing payment system.

2. **Fuel Prize Management**
   - The system will not set or manage fuel prices, it will only calculate the subsidy based on external price data.

<u>Non-Functional Requirements</u>

MUST

1. **Performance**
   - The system must respond to POS eligibility requests within 2 seconds under normal network conditions to ensure smooth user experience at petrol stations.
2. **Availability**
   - The system must have at least 99.5% uptime to support continuous nationwide fuel transactions.
3. **Security and Data Protection**
   - All citizen and transaction data must be encrypted in transit (HTTPS/TLS 1.2 or higher). System must comply with PDPA Malaysia standards.
4. **Authentication and Authorization**
   - All users, admins, and POS terminals must authenticate by using secure tokens. Role-Based Access Control (RBAC) must be enforced.

SHOULD

1. **Maintainability**
   - The system should be modular, with well-documented APIs and microserviced-based architecture for easy updates or fixes.
2. **Interoperability**
   - The system should be designed to integrate with existing government databases.

COULD

1. **Usability**
   - The admin dashboard could follow UX/UI best practices for accessibility and ease of use.

WON'T

1. **Mobile App Optimization**
   - The current version will not focus on mobile app performance tuning, as the main interface is POS and web-based.
2. **Cross-Country Support**
   - The system will not support citizens or fuel stations outside Malaysia in this version.

1.1.2 Use Case Diagram

1. Citizen Registration and Verification

Primary Actor: Citizen (via mobile app / web portal)

Preconditions:

- Citizen has a valid national ID (MyKad) and driver's license.
- Identity Provider API is reachable.
- Citizen is authenticated to the registration portal over TLS.

Main Flow:

1. Citizen opens the official registration portal and enters National ID and Driver's License number.
2. System performs initial format validation and checks for existing profile.
3. System sends verification request to Identity Provider (MyID) to confirm citizen identity. a. If MyID requires multi-factor, the flow forwards to MyID's MFA (OTP/biometrics).
4. Upon successful identity check, System sends a validation request to the Licensing Authority to verify the Driver's License status.
5. Validation Check:

   a. If Identity OR License validation fails: Return specific error to citizen and terminate flow.

   b. If both are successful: System returns a secure identity token and license validity proof.

6. Backend creates or updates Citizen Profile:

   a. Store encrypted PII, set subsidy category, default quotas.

   b. If citizen has prior disqualifying flags, mark for manual review.

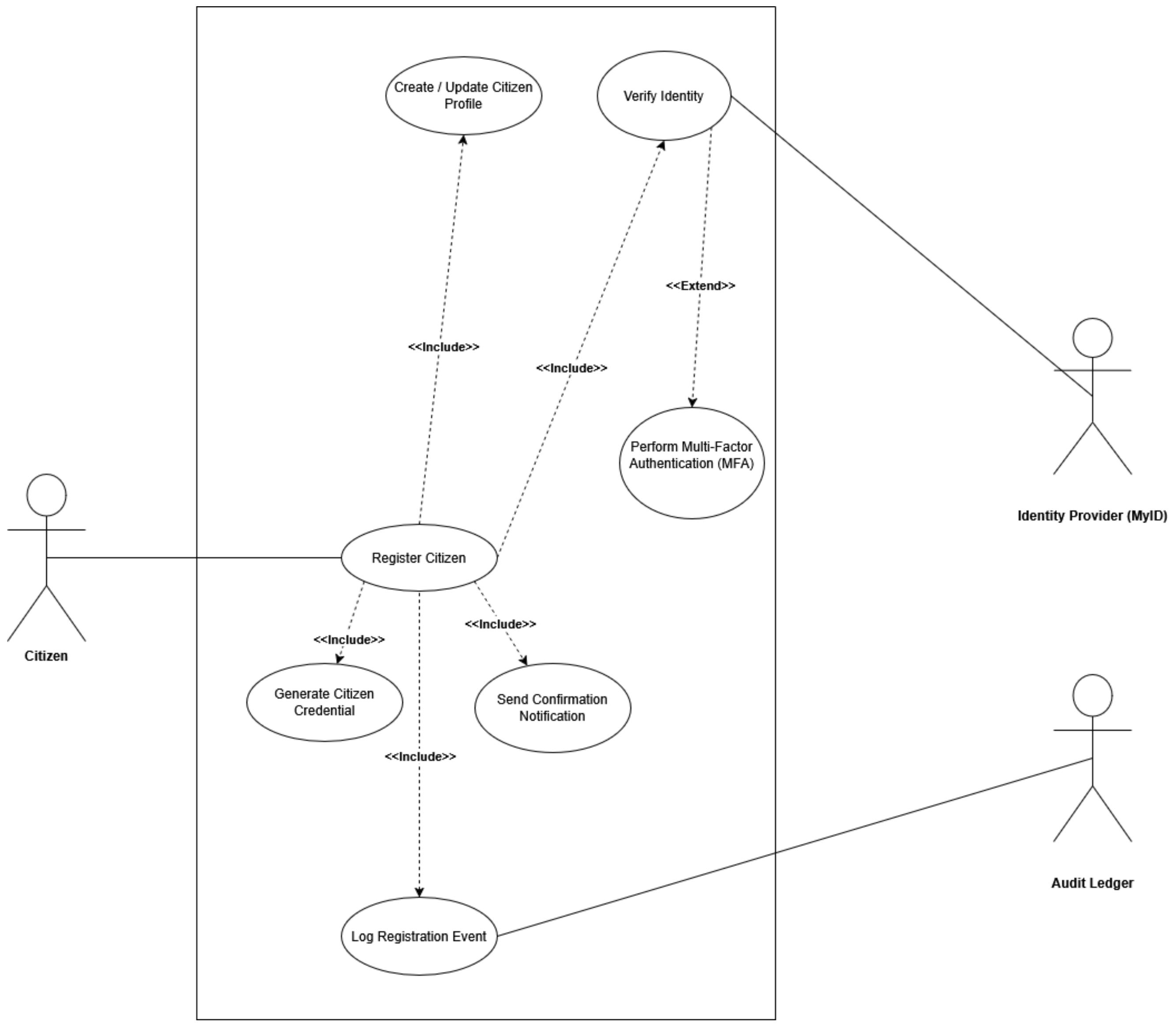
7. Generate a citizen-accessible credential for POS.
8. Send confirmation to citizen via SMS/email with basic subsidy summary and guidance.
9. Log registration events to audit ledger (signed).
10. Return success to citizen.

Alternative Flow:

- Citizen fails MFA → Allow retry; after N attempts lock registration and require manual verification.
- Citizen lacks mobile or cannot receive OTP → Provide in-branch registration at authorized government counter.

Postconditions:

- CitizenProfile exists and is verified or flagged for review.
- Citizen holds a valid token for POS subsidy checks or is notified of next steps.



2. Purchase Fuel with Eligibility Check

Primary Actor: POS Terminal (operated by Gas Station Attendant)

Preconditions:

- Citizen is registered and has a valid subsidy token.
- POS has valid certificate and is authenticated.
- POS and backend clocks are synchronized (NTP).
- Quota data cached recently or available to backend.

Goals:

- The POS receives immediate eligibility response and authorization token. Transaction is committed automically with subsidy applied and recorded for audit and reconciliation.

Main Flow:

1. Attendant enters purchase details on POS: citizen ID, vehicle plate, fuel type and liters or amount.
2. API Gateway authenticates POS and validates client permissions and rate limits.
3. Subsidy Engine retrieves citizen quota from cache.
4. Subsidy Engine responds with eligible:true/false, allowed_liters, subsidy_amount, remaining_quota, eligibility_code, and an authorization_token to be used in commit.
5. POS displays result to attendant showing subsidy split and allowed liters.
6. Attendant dispenses fuel. Once dispensing complete, POS calls POST /api/v1/transactions/commit with authorization_token, actual liters dispensed, final amount.
7. Transaction Service validates authorization token, finalizes transaction, computes subsidy and co-pay, persists transaction record, updates quotas atomically.
8. Payment Gateway is invoked only for the co-pay portion or system issues settlement to station for subsidised portion.
9. Audit Ledger records both eligibility check and commit with cryptographic signature.
10. System returns success to POS with tx_id, final amounts, and receipts info. POS prints/ displays the receipt.
11. Reconciliation events are emitted to event bus for clearing and settlement.

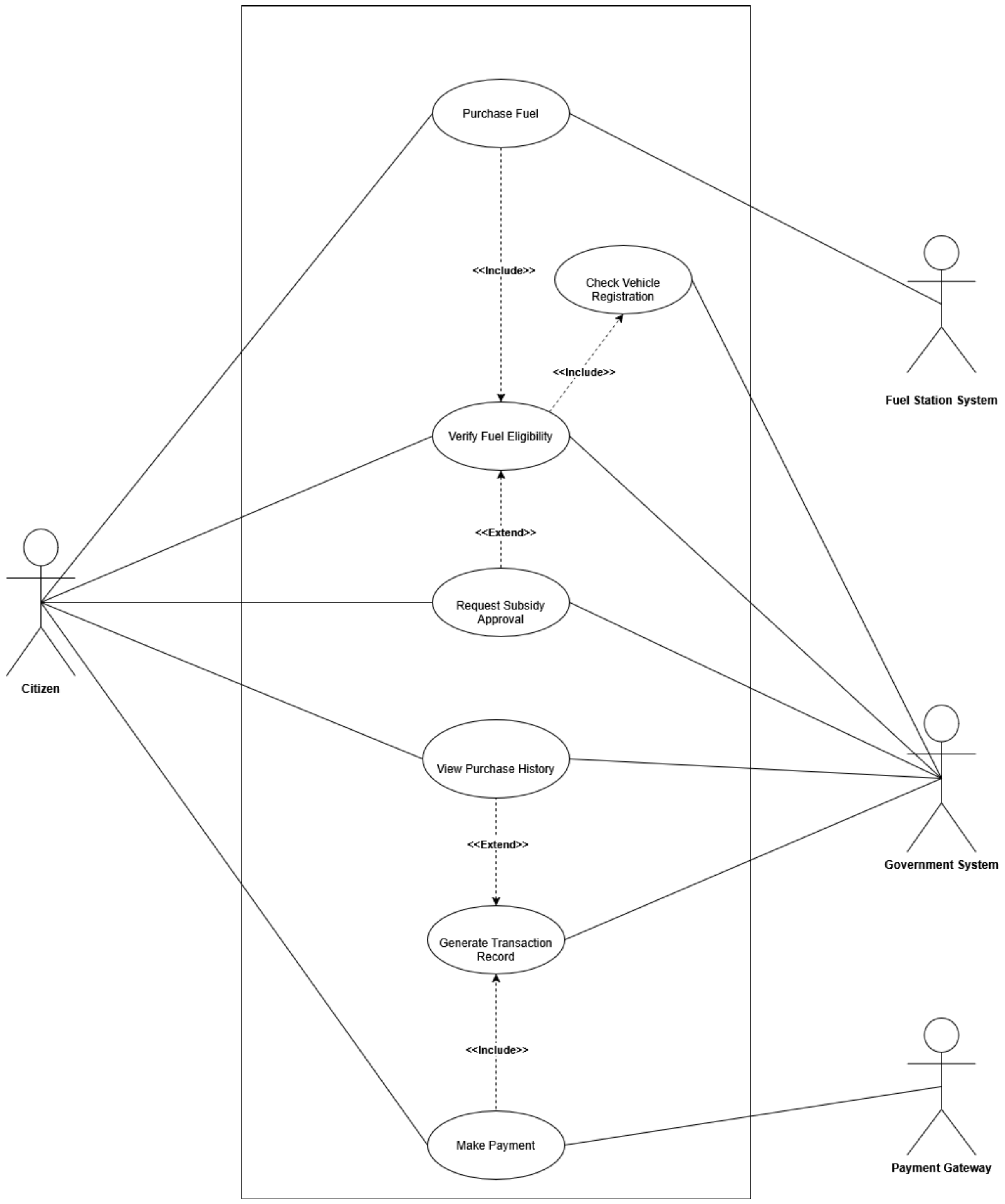
Alternative Flows:

- Eligibility check returns eligible: false → POS displays reason code and aborts subsidised transaction.
- Authorization token expires before commit → POS must re-check eligibility. System may allow a small grace period defined in business rules.

- Quota race condition → Subsidy Engine serializes or uses optimistic concurrency to ensure quotas not overspent; in conflict, second commit denied with clear error.

Postconditions:

- Transaction persisted with subsidy split. Citizen quota updated (reduced).
- Audit ledger contains eligibility check and commits entries.
- Settlement events available for finance systems

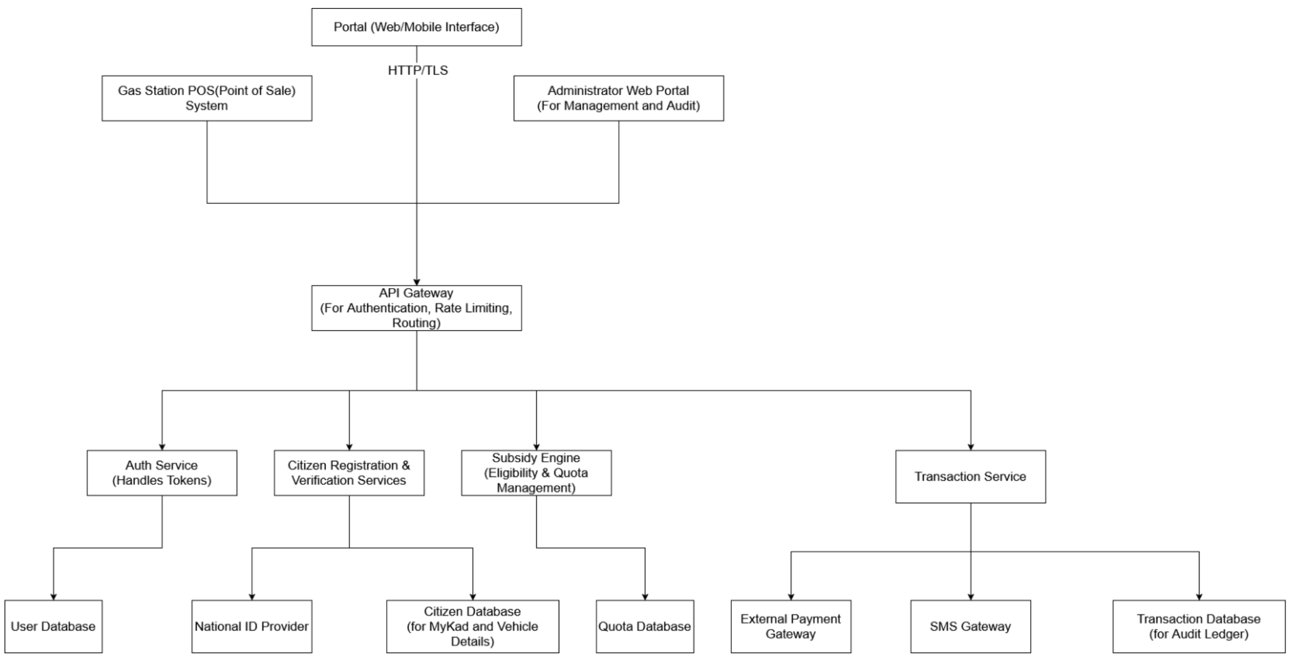


1.2 SYSTEM ARCHITECTURE AND DESIGN

1.2.1 High-Level System Architecture Diagram

The proposed high-level system architecture, characterized by its segmentation into distinct services such as Auth Service, Subsidy Engine, and Transaction Service is specifically justified by the need to meet the critical Non-Functional Requirements (NFRs) for Scalability, Security, and Reliability.

The Microservices architecture allows for the independent development, deployment, and scaling of these distinct business capabilities, which is essential for a nationwide, high-volume, and security-critical system.



Portal (Web/Mobile Interface)
HTTP/TLS
Gas Station POS(Point of Sale) System
Administrator Web Portal (For Management and Audit)
API Gateway (For Authentication, Rate Limiting, Routing)
Auth Service (Handles Tokens)
Citizen Registration & Verification Services
Subsidy Engine (Eligibility & Quota Management)
Transaction Service
User Database
National ID Provider
Citizen Database (for MyKad and Vehicle Details)
Quota Database
External Payment Gateway
SMS Gateway
Transaction Database (for Audit Ledger)

Justification based on Scalability

The architecture is designed to meet the requirement for high performance, specifically the 2-second response time for POS eligibility requests.

Independent Scaling: The most critical component, the Subsidy Engine (which handles eligibility and quota checks against the Quota Database), can be scaled horizontally (adding more instances) independent of lower-traffic services (like the Administrator Web Portal or user registration).

Decoupled Workloads: High demand on one area (e.g., peak-hour transactions) will not degrade the performance of another, ensuring that the critical 2-second response time is consistently maintained across the network.

API Gateway Rate Limiting: The API Gateway provides a central point to manage and protect the backend services from unexpected traffic surges, ensuring resource availability for legitimate, real-time POS requests.

Justification based on Security

The architecture provides a robust security posture necessary for handling sensitive citizen and transaction data and complying with PDPA Malaysia standards.

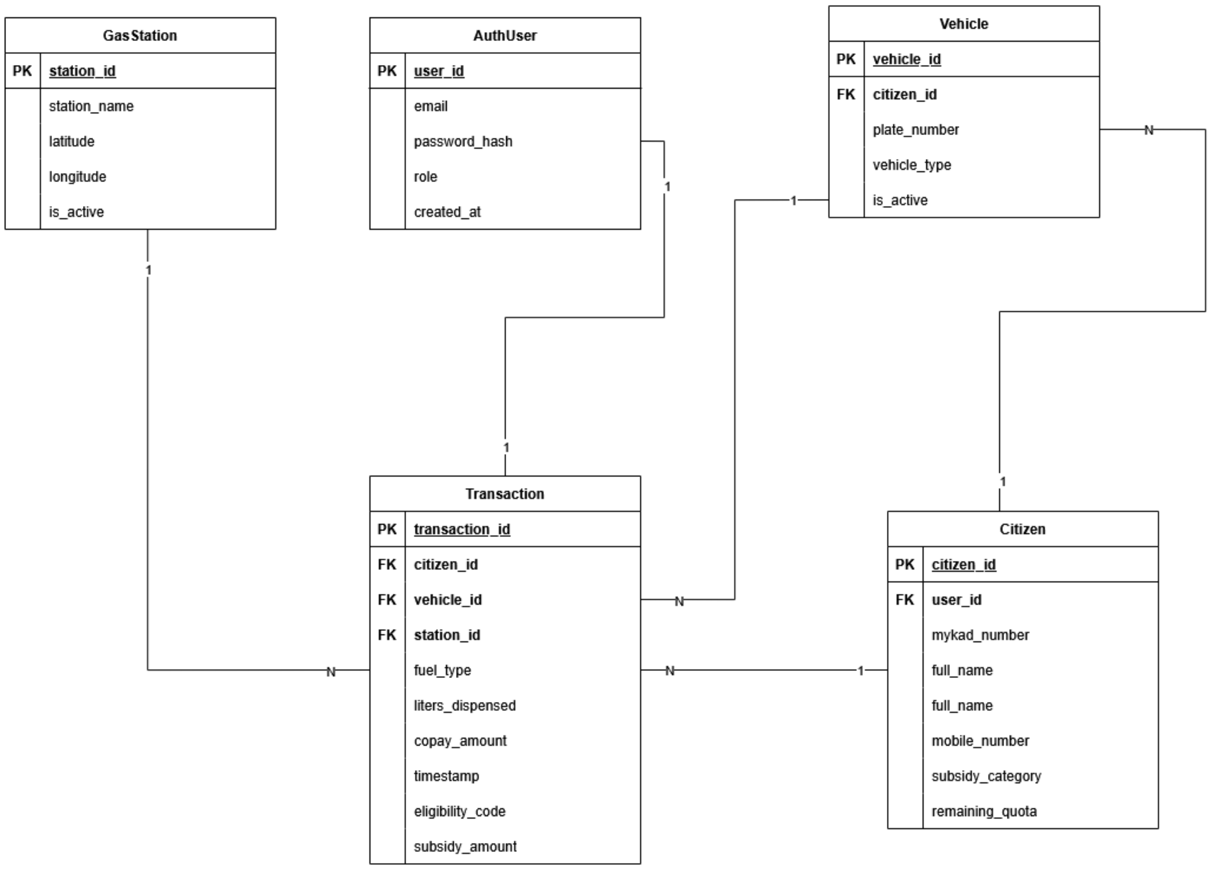
- **Centralized Authentication and RBAC:** The Auth Service is the single source of truth for all access control. All requests from the POS System, Portal, and Administrator must validate their secure tokens at the API Gateway before being routed. This central enforcement point ensures strict Role-Based Access Control (RBAC) is applied system-wide.
- **Data Segregation and Isolation:** Sensitive data is segmented across distinct databases (User Database, Citizen Database, Transaction Database). A breach or vulnerability in a peripheral service cannot automatically access all sensitive data, minimizing the "blast radius" and facilitating targeted compliance with PDPA data handling requirements.
- **In-Transit Encryption:** The use of HTTP/TLS from the Portal to the API Gateway ensures all communication, including citizen and transaction data, is encrypted in transit (TLS 1.2 or higher), directly fulfilling the data protection requirement.

Justification based on Reliability

The distributed nature of the microservices architecture is the primary mechanism for achieving the **MUST** requirement of **99.5% uptime**.

- **Fault Isolation:** Failures are contained within the boundary of a single service. For example, if the SMS Gateway or the Citizen Registration Service experiences an outage, the core, mission-critical workflow—the Transaction Service and Subsidy Engine—will continue operating.
- **Independent Deployment (Maintainability):** In line with the SHOULD requirement for modularity, updates or bug fixes to a single service (e.g., a fix to the Citizen Registration & Verification Services) can be deployed without requiring a system-wide reboot or downtime. This minimizes maintenance windows, directly supporting high Availability.
- **Audit Ledger:** The dedicated Transaction Database (for Audit Ledger) ensures that a verifiable record of every transaction is maintained, even during temporary failures in other systems, safeguarding data integrity and regulatory compliance.

1.2.2 Detailed Database Schema



The schema is structured around five primary entities: GasStation, AuthUser, Citizen, Vehicle, and Transaction.

1. GasStation

This table manages the physical locations where transactions occur.

Field Name	Type/Role	Purpose
station_id	PK (Primary Key)	Unique identifier for each gas station.
station_name	Data Field	The registered name of the gas station.
latitude, longitude	Data Field	Geographical coordinates for location tracking and potential compliance checking.
is_active	Data Field	Status flag to enable or disable the station's participation in the subsidy system.
Relationship	1-to-N with Transaction	One gas station can process many transactions.

2. AuthUser

This table manages login credentials and roles for system users (e.g., Administrators, potentially Gas Station operators). This is separate from the *Citizen* data for security.

Field Name	Type/Role	Purpose
user_id	PK (Primary Key)	Unique identifier for system login accounts.
email	Data Field	Used as the login identifier.
password_hash	Data Field	Stores the securely hashed password (critical for security compliance).
role	Data Field	Defines the user's access level (e.g., 'admin', 'auditor'), enforcing RBAC.
created_at	Data Field	Timestamp of user creation.
Relationship	1-to-1 with Citizen	A system login account can be linked to a single citizen profile (if the citizen is also a portal user).

3. Citizen

This is the core table storing subsidy beneficiary profiles and their entitlements.

Field Name	Type/Role	Purpose
citizen_id	PK (Primary Key)	Unique identifier for the subsidy beneficiary.
user_id	FK (Foreign Key)	Links to the AuthUser table.
mykad_number	Data Field	The unique national ID number, critical for verification and auditing.
full_name, mobile_number	Data Field	Contact and identification details.
subsidy_category	Data Field	Determines the fuel eligibility and quota rules for the citizen.
remaining_quota	Data Field	The crucial, real-time value representing the subsidy liters available to the citizen.
Relationship	1-to-N with Vehicle	One citizen can own or register multiple vehicles for subsidy.
Relationship	1-to-N with Transaction	One citizen can be involved in many transactions.

4. Vehicle

This table records vehicle details and their registration under a citizen for subsidy purposes.

Field Name	Type/Role	Purpose
vehicle_id	PK (Primary Key)	Unique identifier for the vehicle.
citizen_id	FK (Foreign Key)	Links the vehicle to its registered owner/subsidy beneficiary.
plate_number	Data Field	The vehicle registration number (license plate).
vehicle_type	Data Field	Vehicle classification (e.g., 'motorcycle', 'sedan', 'truck'), which may affect subsidy rules.
is_active	Data Field	Status flag to determine if the vehicle is currently eligible for subsidy.
Relationship	1-to-N with Transaction	One vehicle can be used in many transactions.

5. Transaction

This is the central ledger for all subsidy events, recording the details of every fuel purchase.

Field Name	Type/Role	Purpose
transaction_id	PK (Primary Key)	Unique identifier for each transaction record.
citizen_id, vehicle_id, station_id	FK (Foreign Keys)	Links the transaction to the specific beneficiary, vehicle, and gas station.
fuel_type	Data Field	Type of fuel dispensed (e.g., RON95, Diesel).
liters_dispensed	Data Field	Volume of fuel purchased.
copay_amount	Data Field	The portion of the cost paid by the user (non-subsidized amount).
timestamp	Data Field	Record of when the transaction occurred (critical for auditing).
eligibility_code	Data Field	Code returned by the Subsidy Engine (e.g., success, quota exceeded, invalid vehicle).
subsidy_amount	Data Field	The value of the subsidy applied to the transaction.
Relationship	N-to-1	Links back to Citizen, Vehicle, and GasStation.

Schema Efficiency and Support for System Requirements

- **Efficient Quota Management:** The remaining_quota field is stored directly in the Citizen table, allowing the Subsidy Engine to perform fast, single-row lookups and updates to check and deduct the quota in real-time. This is crucial for meeting the 2-second POS performance requirement.
- **Clear Audit Trail:** The Transaction table acts as the system's ledger, linking every purchase to the who (citizen), what (vehicle), where (station), and when (timestamp). This structure is essential for compliance and auditing.
- **Security Segmentation:** By separating AuthUser (logins/roles) from Citizen (personal/subsidy data), the system enforces a key security principle, protecting beneficiary data from potential exposure in the authentication layer.

1.3 IMPLEMENTATION AND PROTOTYPING

This section outlines the development strategy for the Fuel Subsidy Management System and presents the Proof of Concept (PoC) for the critical eligibility verification module.

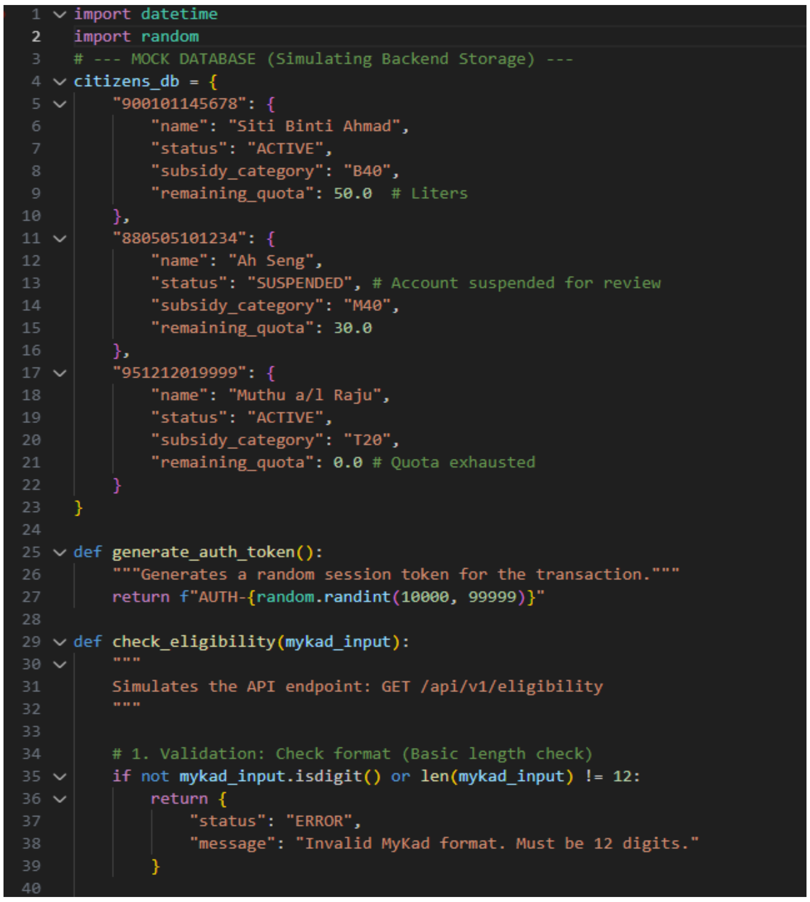
1.3.1 Working Prototype Using Console-Based Verification

A console-based prototype has been developed to demonstrate the core logic of the Citizen Verification Module. This prototype simulates the interaction between a POS terminal and the centralized Subsidy Engine.

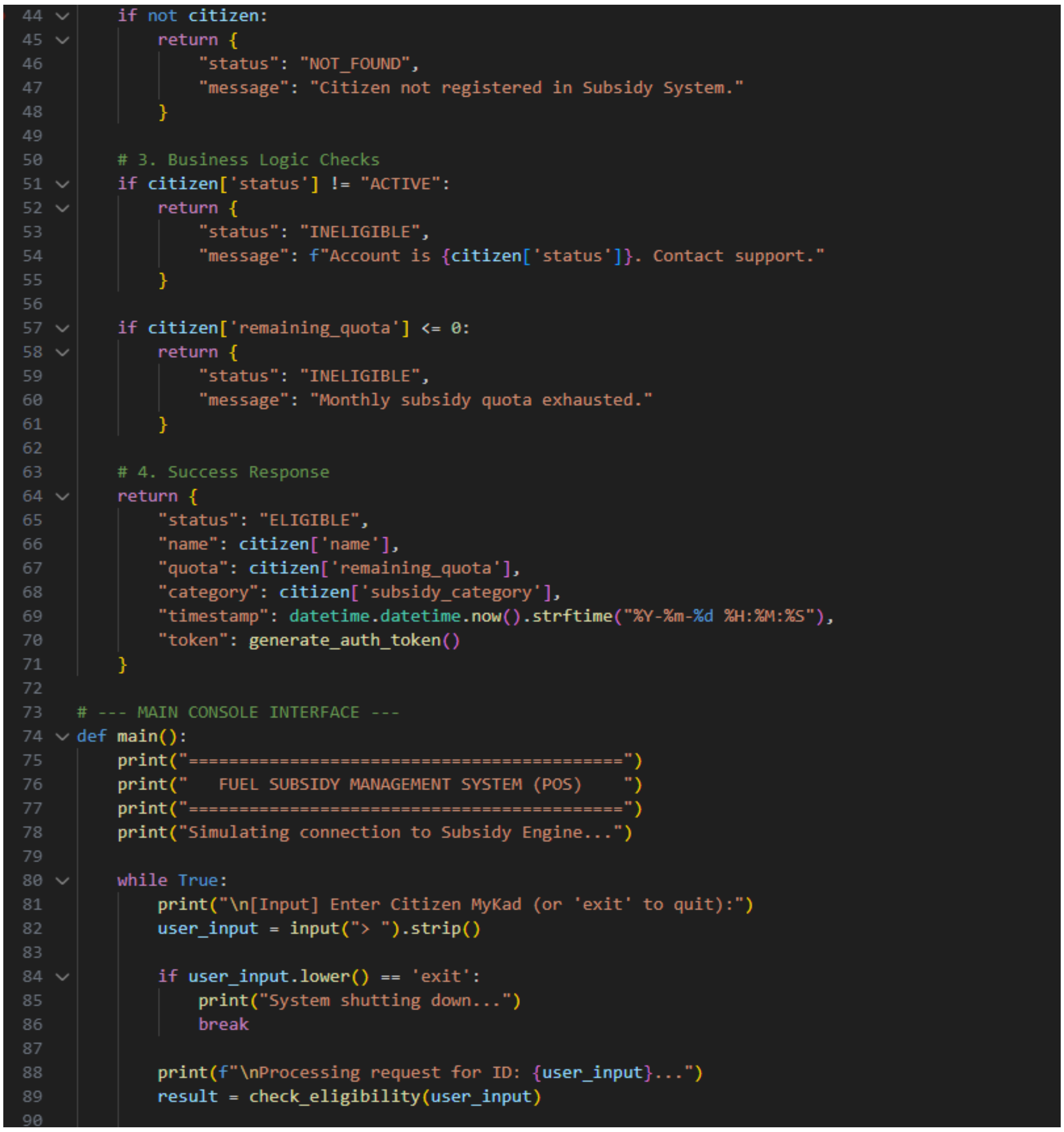
Prototype Functional Scope:

1. **Input:** Accepts a 12-digit MyKad number from the user.
2. **Validation:** Validates input format and checks against a simulated in-memory database.
3. **Logic:** Verification rules include checking if the citizen is "ACTIVE", if they belong to a subsidized category (B40/M40), and if they have remaining_quota > 0.
4. **Output:** Returns an eligibility status (ELIGIBLE or INELIGIBLE) along with relevant details (Name, Quota Balance, Authorization Token).

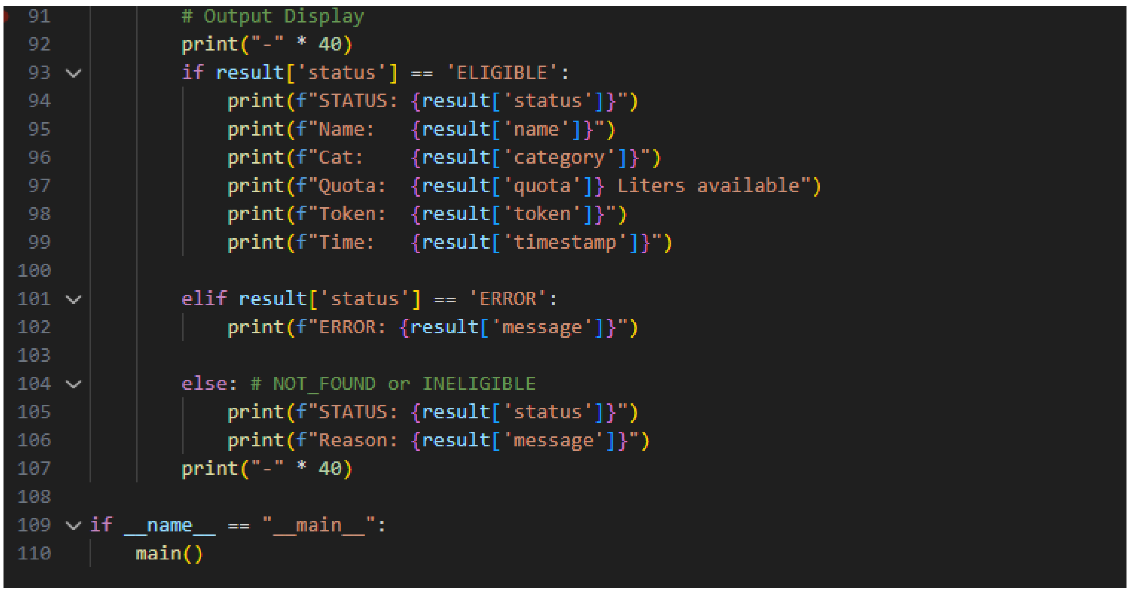
Source Code (Python):



```python
import datetime
import random
# --- MOCK DATABASE (Simulating Backend Storage) ---
citizens_db = {
    "900101145678": {
        "name": "Siti Binti Ahmad",
        "status": "ACTIVE",
        "subsidy_category": "B40",
        "remaining_quota": 50.0  # Liters
    },
    "880505101234": {
        "name": "Ah Seng",
        "status": "SUSPENDED", # Account suspended for review
        "subsidy_category": "M40",
        "remaining_quota": 30.0
    },
    "951212019999": {
        "name": "Muthu a/l Raju",
        "status": "ACTIVE",
        "subsidy_category": "T20",
        "remaining_quota": 0.0 # Quota exhausted
    }
}

def generate_auth_token():
    """Generates a random session token for the transaction."""
    return f"AUTH-{random.randint(10000, 99999)}"

def check_eligibility(mykad_input):
    """
    Simulates the API endpoint: GET /api/v1/eligibility
    """

    # 1. Validation: Check format (Basic length check)
    if not mykad_input.isdigit() or len(mykad_input) != 12:
        return {
            "status": "ERROR",
            "message": "Invalid MyKad format. Must be 12 digits."
        }

```

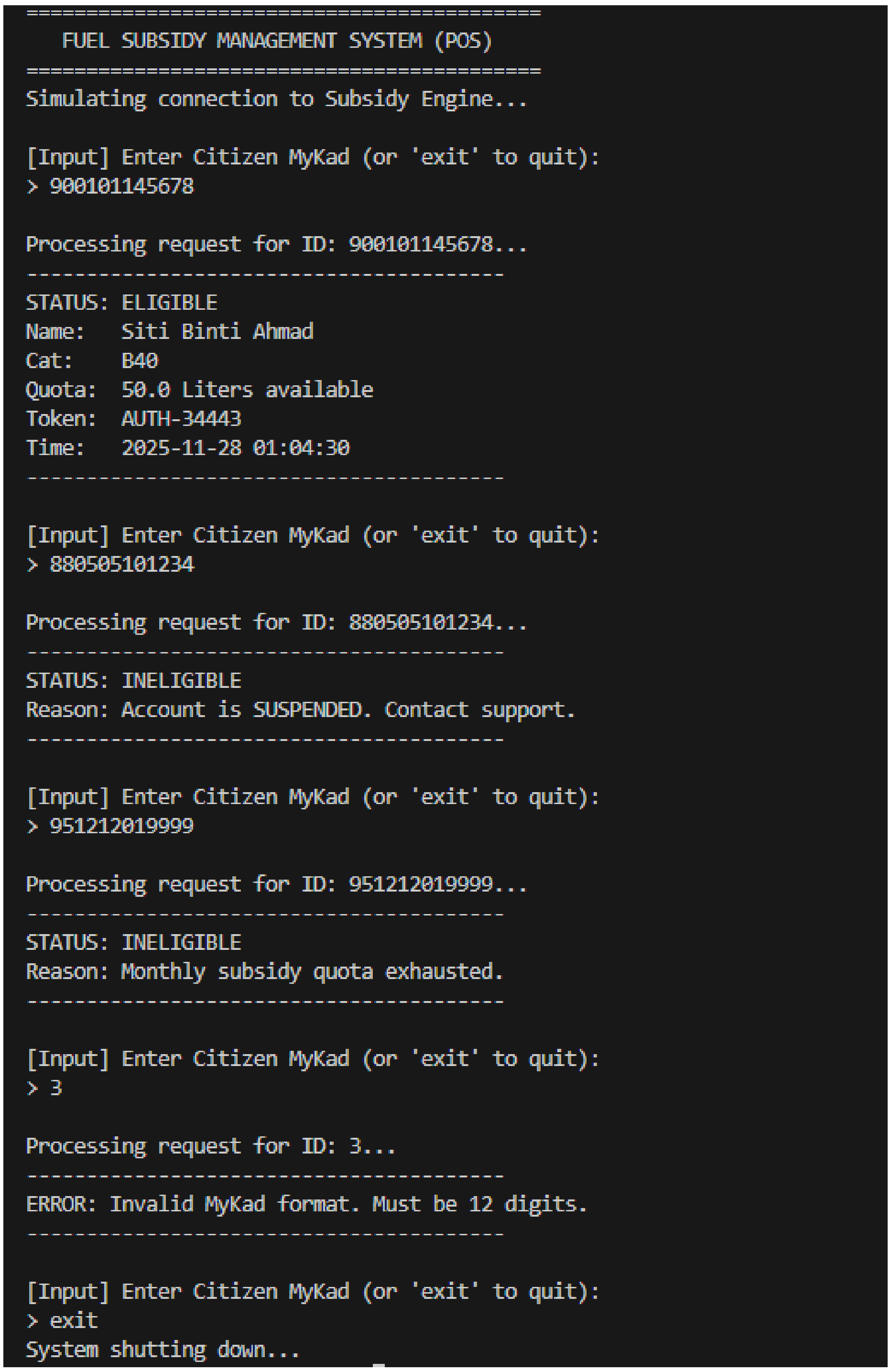


```python
    if not citizen:
        return {
            "status": "NOT_FOUND",
            "message": "Citizen not registered in Subsidy System."
        }

    # 3. Business Logic Checks
    if citizen['status'] != "ACTIVE":
        return {
            "status": "INELIGIBLE",
            "message": f"Account is {citizen['status']}. Contact support."
        }

    if citizen['remaining_quota'] <= 0:
        return {
            "status": "INELIGIBLE",
            "message": "Monthly subsidy quota exhausted."
        }

    # 4. Success Response
    return {
        "status": "ELIGIBLE",
        "name": citizen['name'],
        "quota": citizen['remaining_quota'],
        "category": citizen['subsidy_category'],
        "timestamp": datetime.datetime.now().strftime("%Y-%m-%d %H:%M:%S"),
        "token": generate_auth_token()
    }

# --- MAIN CONSOLE INTERFACE ---
def main():
    print("===========================================")
    print("   FUEL SUBSIDY MANAGEMENT SYSTEM (POS)    ")
    print("===========================================")
    print("Simulating connection to Subsidy Engine...")

    while True:
        print("\n[Input] Enter Citizen MyKad (or 'exit' to quit):")
        user_input = input("> ").strip()

        if user_input.lower() == 'exit':
            print("System shutting down...")
            break

        print(f"\nProcessing request for ID: {user_input}...")
        result = check_eligibility(user_input)

```



```python
        # Output Display
        print("-" * 40)
        if result['status'] == 'ELIGIBLE':
            print(f"STATUS: {result['status']}")
            print(f"Name:   {result['name']}")
            print(f"Cat:    {result['category']}")
            print(f"Quota:  {result['quota']} Liters available")
            print(f"Token:  {result['token']}")
            print(f"Time:   {result['timestamp']}")

        elif result['status'] == 'ERROR':
            print(f"ERROR: {result['message']}")

        else: # NOT_FOUND or INELIGIBLE
            print(f"STATUS: {result['status']}")
            print(f"Reason: {result['message']}")
        print("-" * 40)

if __name__ == "__main__":
    main()
```

Sample Output:



```
============================================
   FUEL SUBSIDY MANAGEMENT SYSTEM (POS)
============================================
Simulating connection to Subsidy Engine...

[Input] Enter Citizen MyKad (or 'exit' to quit):
> 900101145678

Processing request for ID: 900101145678...
----------------------------------------
STATUS: ELIGIBLE
Name:   Siti Binti Ahmad
Cat:    B40
Quota:  50.0 Liters available
Token:  AUTH-34443
Time:   2025-11-28 01:04:30
----------------------------------------

[Input] Enter Citizen MyKad (or 'exit' to quit):
> 880505101234

Processing request for ID: 880505101234...
----------------------------------------
STATUS: INELIGIBLE
Reason: Account is SUSPENDED. Contact support.
----------------------------------------

[Input] Enter Citizen MyKad (or 'exit' to quit):
> 951212019999

Processing request for ID: 951212019999...
----------------------------------------
STATUS: INELIGIBLE
Reason: Monthly subsidy quota exhausted.
----------------------------------------

[Input] Enter Citizen MyKad (or 'exit' to quit):
> 3

Processing request for ID: 3...
----------------------------------------
ERROR: Invalid MyKad format. Must be 12 digits.
----------------------------------------

[Input] Enter Citizen MyKad (or 'exit' to quit):
> exit
System shutting down...
```

1.3.2 Fraud Detection Rule Pseudocode

This pseudocode is to prevent "impossible travel" fraud where a single ID is used at two geographically distant locations in a short time. The system employs a velocity check algorithm.

Rule Definition: Flag any transaction if the distance from the previous transaction is > 100km AND the time difference is < 1 hour.

Pseudocode:

```
1.0 START
1.1 GET currentTransaction details (citizen_ID, station_ID, currentTimestamp)
1.2 RETRIEVE previousTransaction FROM Database WHERE citizen_ID
currentTransaction.citizen_ID ORDER BY timestamp DESC LIMIT 1
1.3 IF previousTransaction EXISTS THEN
  1.3.1 CALCULATE distanceKm = getDistance(station_ID,
        previousTransaction.location)
  1.3.2 CALCULATE timeDiffMinutes = currentTransaction.timestamp -
        previousTransaction.timestamp
  1.3.3 IF distanceKm > 100 AND timeDiffMinutes < 60 THEN
    FLAG currentTransaction AS "Potential Fraud"
    LOG ALERT "Velocity Rule Violation" with citizen_ID
    DISPLAY "Fraud Detected."
  ELSE
    RETURN “No Fraud Detected.”
  ENDIF
1.4 ELSE
  DISPLAY "Previous Transaction Missing. Recording Current Transaction..."
  PROMPT and GET currentTransaction
1.5 STORE currentTransaction INTO database FOR future comparison
2.0 END
```

1.4 TESTING AND QUALITY ASSURANCE

1.4.1 Test Plan

Testing is carried out across two major levels: unit testing to verify the correctness of individual modules, and integration testing to ensure coordinated behaviour across the full transaction workflow.

A. Unit Testing (Component-Level)

Unit tests validate each backend module independently to ensure correctness, proper error handling, and data consistency.

1. Unit Tests for Verification Module (/verifyCitizen)

Test Case	Description	Expected Output
Valid MyKad Lookup	Lookup using existing MyKad	Eligible = true, correct citizen data returned
Non-existent Citizen	MyKad not found	eligible = false, CITIZEN_NOT_FOUND
Invalid MyKad Format	MyKad with wrong length or symbols	VALIDATION_ERROR
Citizen With Zero Quota	remaining_quota = 0	eligible = false, QUOTA_EXCEEDED
Authorization Token Generation	Valid citizen data	Unique, random, time-limited token generated

2. Unit Tests for Subsidy Engine

Test Case	Description/Logic Tested	Expected Output
Correct Quota Deduction	Deduct 10L from remaining quota	Quota updated accurately
Disallowed Fuel Type	Citizen attempts non-subsidized fuel	INVALID_FUEL_TYPE
Allowed Liters Calculation	Requested liters > remaining quota	allowed_liters = remaining_quota
Fraud Detection Trigger	2 stations >100km apart within 1 hour	fraud_flag = true

B. Integration Testing (System-Level)

Integration tests validate the full workflow across multiple modules: Verification → Subsidy Engine → Transaction Service → Database.

1. **Complete Successful Transaction**
   - *Flow:* Verify citizen → calculate eligibility → generate token → commit transaction.
   - *Expected:*
     - Transaction saved in database
     - Quota updated atomically
     - Receipt info returned to POS
2. **Token Expiry Scenario**
   - *Flow:* Generate token → wait until expiry → attempt commit.
   - *Expected:*
     - TOKEN_EXPIRED; POS must re-check eligibility
3. **Quota Race Condition**
   - *Flow:* Two commit requests arrive at the same time.
   - *Expected:*
     - Only first commit succeeds
     - Second commit returns QUOTA_CONFLICT
4. **Integration with Fraud Detection**
   - *Flow:* Two far-apart stations within 60 mins
   - *Expected:*
     - System sets fraud_flag on second transaction
5. **Insufficient Quota at Commit Time**
   - *Flow:* Eligibility OK at first check → quota updated by another transaction → commit.
   - *Expected:*
     - Commit rejected and quota remains consistent.

1.4.2 Security Threat Analysis & Mitigation Strategies

To ensure the system remains secure against common and high-impact threats, three major risks were identified along with their mitigation strategies. The table below organizes these threats for clarity.

Threat 1: Data Breach (Unauthorized Access to Citizen/Transaction Data)

Risk: Attackers gaining access to MyKad numbers, subsidy amounts, or transaction history. Mitigation:

- Enforce TLS 1.2+ for all communications.
- Encrypt sensitive data at rest (MyKad, mobile number).
- Apply RBAC with least-privilege access for admins, auditors, and terminals.
- Use segregated databases (AuthUser separate from Citizen).
- Implement periodic security audits and database access logging.

Threat 2: QR Code / Token Spoofing at POS

Risk: Attackers generate fake subsidy tokens or alter QR data. Mitigation:

- All authorization tokens must be digitally signed (JWT with signature).
- POS must validate token signature via Auth Service.
- Tokens time-limited (e.g., valid for 5 minutes only).
- QR code encodes only token, not personal data—reduces exposure.
- Rate limit repeated attempts from the same POS device.

Threat 3: Replay Attacks During Eligibility Check or Commit

Risk: Attacker replays old eligibility responses or commit requests to claim additional subsidy. Mitigation:

- Add nonce (unique request ID) to every eligibility check and commit request.
- Server rejects reused nonces.
- All commit operations tied to a single-use authorization token.
- Audit ledger with cryptographic signatures ensures traceability.

SECTION 2: HUMAN-COMPUTER INTERACTION

2.1 USER RESEARCH AND ANALYSIS

2.1.1 Persona Development

Persona A – Citizen User: Siti

Profile Summary: Siti is a 60-year-old primary school teacher who relies on her car for her daily commute but feels anxious about adopting the new technology. While she is capable of using basic communication apps like WhatsApp, she lacks confidence with administrative system and struggles with small screens due to declining eyesight and slight hand tremors.

Goals:

- Successfully register for the subsidy without complications.
- Verify her eligibility status clearly.
- Complete the registration form without errors.

Limitations:

- Small font sizes that are hard to read without reading glasses.
- Unclear error messages that don't explain how to fix the mistakes.
- Fear of "breaking" the system.

Persona B – Gas Station Attendant: Adam

Profile Summary: Adam is a 22-year-old "digital native" working as a gas station during the high-pressure morning rush hour. He is highly proficient with technology and quick to learn new systems, but he is hampered by his environment specifically outdoor screen glare, noise and impatient customers.

Goals:

- Process subsidy verifications under 15 seconds.
- Avoid manual entry unless absolutely necessary.
- Minimize arguments with customers regarding system failure.

Limitations:

- Laggy interfaces that slow down the queue.
- Too many "taps" required to complete a single transaction.
- Screens that are not readable in direct sunlight.

2.1.2 Task Analysis

Step-by-Step Task Breakdown for Siti Registering the Subsidy Process:

1. Locate and tap the specific subsidy icon on the smartphone home screen.
2. Select the "Create New Account" button on the landing page.
3. Type in full name, MyKad identification number, and phone number into the provided fields.
4. Capture or select a clear photo of the MyKad for identify validation.
5. Tap the "Submit" button to send the form to the database.
6. Receive a notification or SMS confirming the account status.
7. Log back into the dashboard to view the unique QR Code required for claiming fuel.

Top 3 Potential Pain Points

- **The "Document Upload" Friction**

  Siti does not know where her files are stored on her phone. Taking a live photo of her ID is difficult because her hands shake slightly, resulting in blurry photos that the system rejects without a clear reason.

- **Small Touch Targets & Fonts**

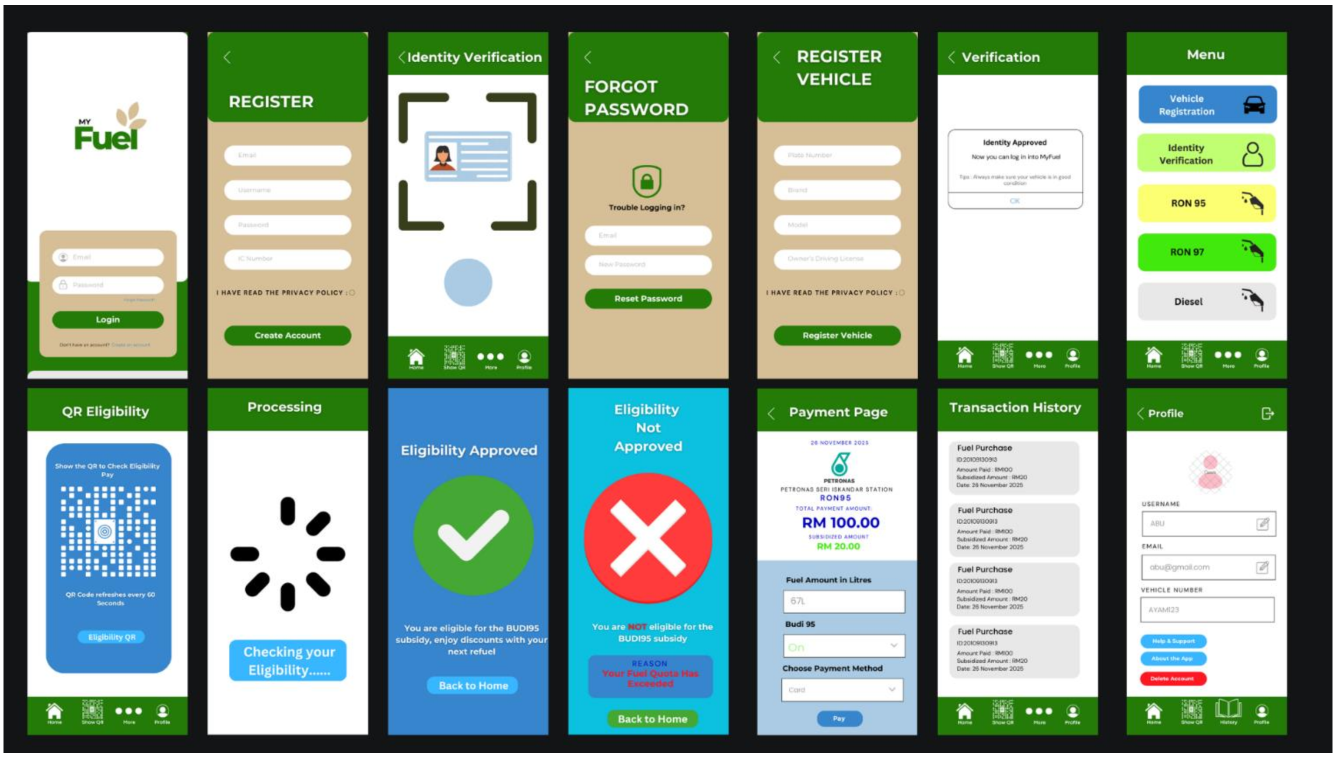
  Standard 12px text and small input fields are hard for her to focus on. If the "Submit" button is too close to a "Cancel" or "Back" button, she risks tapping the wrong one due to reduced motor control.

- **Cognitive Load of OTPs**

  When the OTP arrives, she has to switch apps from the subsidy app to the SMS application. She often forgets the 6-digit code in the 3 seconds it takes to switch back or she struggles to navigate back to the app where she started.

2.2 INTERFACE DESIGN AND PROTOTYPING

2.2.1 CITIZEN MOBILE APP – LOW FIDELITY WIREFRAME



The purpose of the Low Fidelity Wireframe UI is to meet the citizen's requirements which is an intuitive mobile interface with low fidelity which was developed using Canva. The design focuses on a smooth flow of using the application with cor features being displaying the QR code for eligibility checking, mobile fuel purchase page and a main menu with various options.

<u>**Included Pages**</u>

1. Login / Registration / Forget Password / Profile pages

- Clear fields to be filled in or displayed .
- Straightforward flow to prevent confusion.

2. Main Menu

- Clearly display various options which are available for citizens to use.
- Emphasis on icons and colors for different option types to improve user comprehension.

3. QR Code Display Screen

- Large, scannable QR code quick scanning.
- Shows subsidy eligibility.

4.Payment Page

- Clearly displays payment information with large Icons and contrasting colors.

5.Transaction History Page

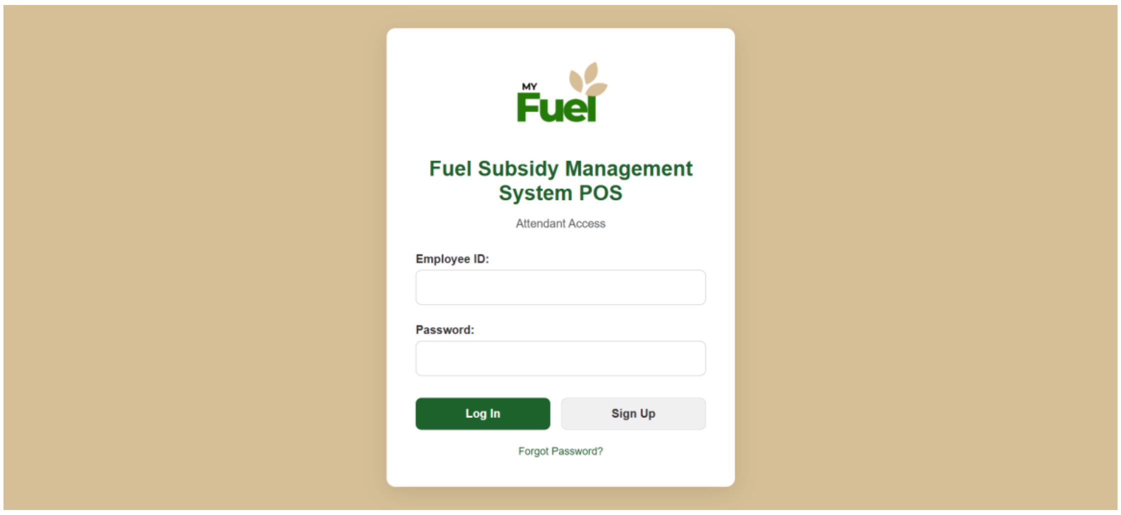
- Clearly displays transactions history with all the details.

Design Rationale

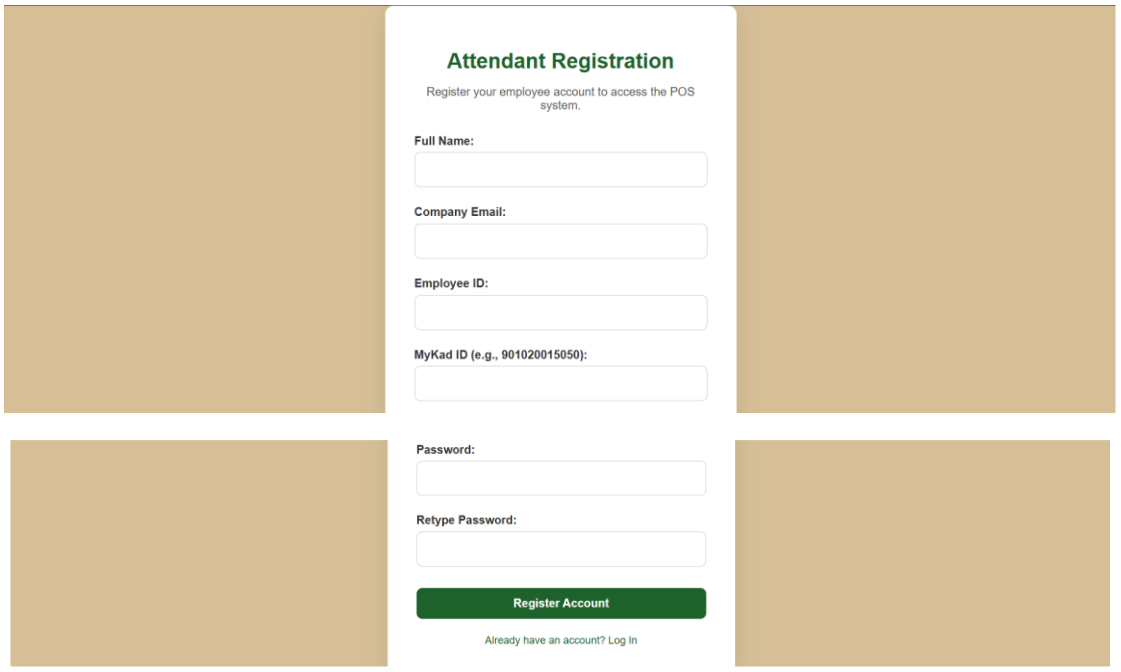
- Prioritizes simple and clear design for unfamiliar users.
- Designed with linear navigation in mind to avoid confusion.
- Follows design principles which are consistency, visibility and simplicity.

2.2.2 ATTENDANT'S POS INTERFACE – HIGH FIDELITY PROTOTYPE

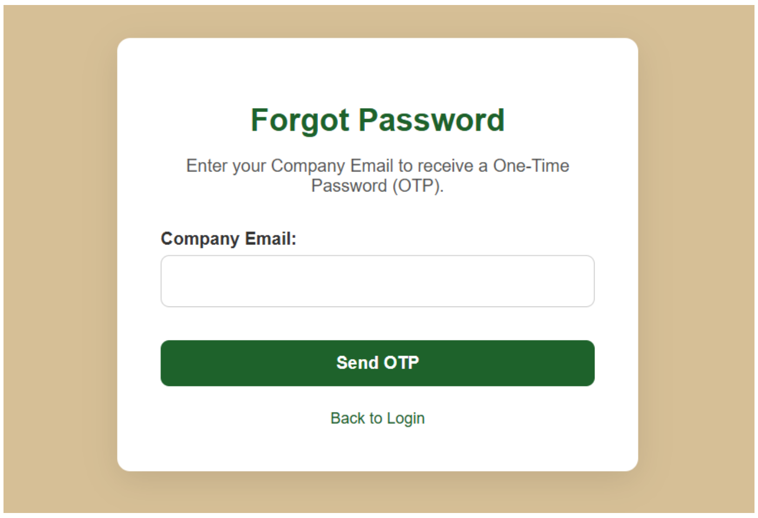
Login Page:



Registration Page:


Attendant Registration
Register your employee account to access the POS system.
Full Name:
Company Email:
Employee ID:
MyKad ID (e.g., 901020015050):
Password:
Retype Password:
Register Account
Already have an account? Log In

Forgot Password Page:



Forgot Password

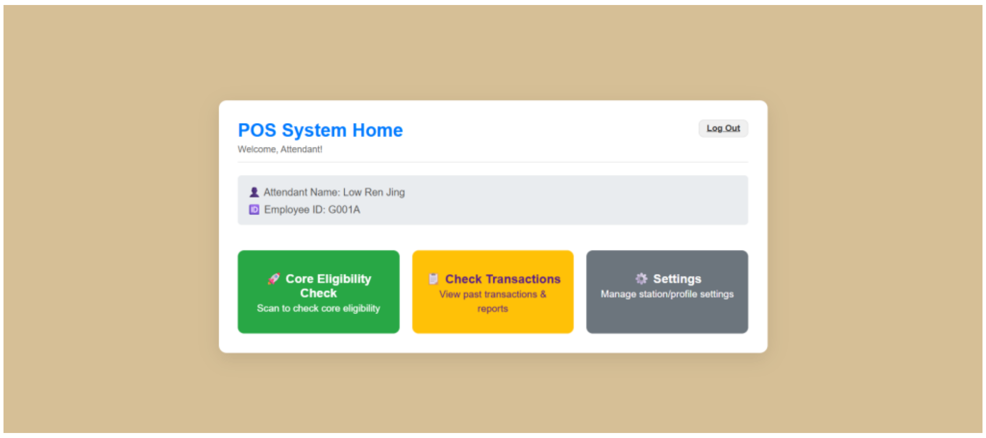
Enter your Company Email to receive a One-Time Password (OTP).

Company Email:

Send OTP

Back to Login

Home Page:



POS System Home

Welcome, Attendant!

Log Out

Attendant Name: Low Ren Jing

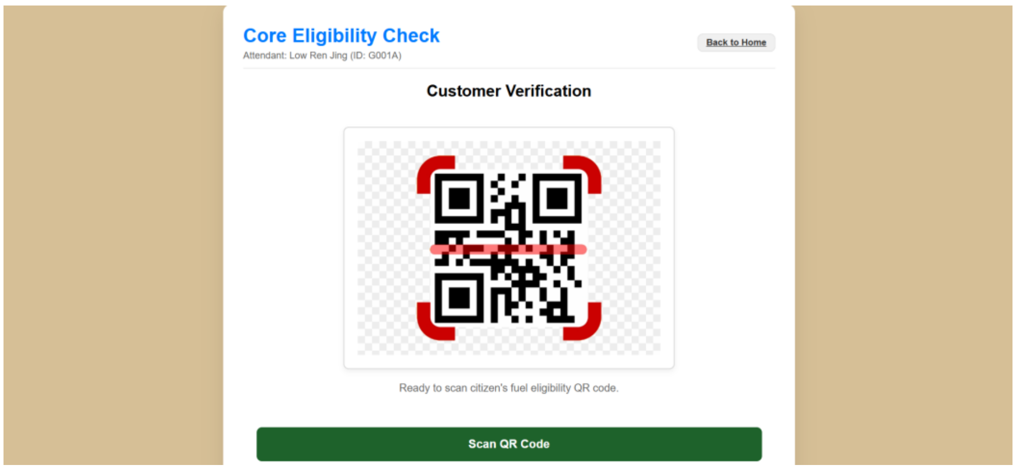
Employee ID: G001A

Core Eligibility Check
Scan to check core eligibility

Check Transactions
View past transactions & reports

Settings
Manage station/profile settings

Check Eligibility Page:



Core Eligibility Check

Attendant: Low Ren Jing (ID: G001A)

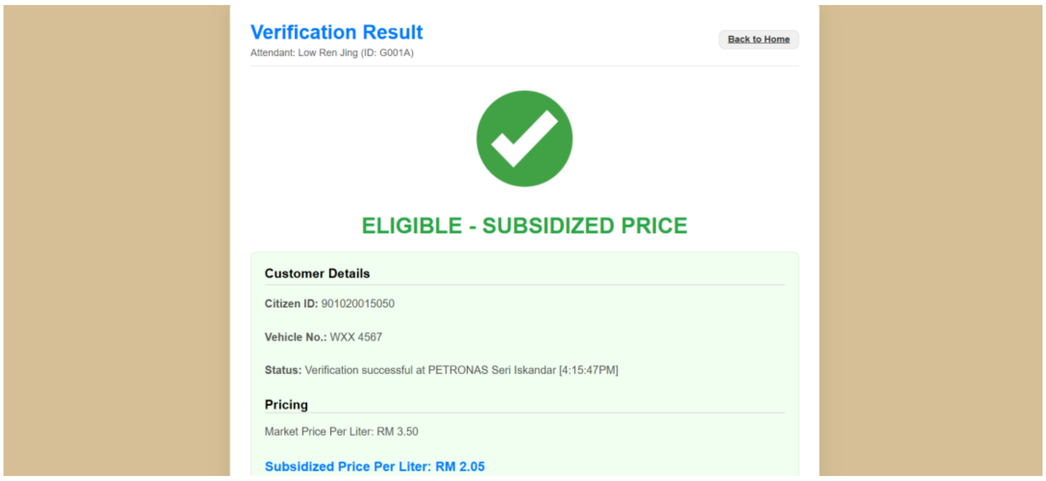
Back to Home

Customer Verification

Ready to scan citizen's fuel eligibility QR code.

Scan QR Code

Verification Result Page:



Verification Result

Attendant: Low Ren Jing (ID: G001A)

Back to Home

ELIGIBLE - SUBSIDIZED PRICE

Customer Details

Citizen ID: 901020015050

Vehicle No.: WXX 4567

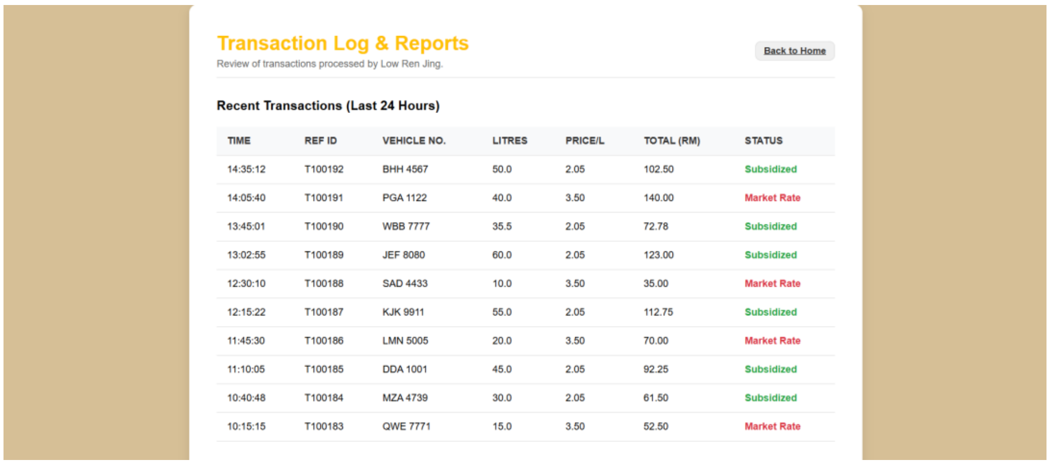
Status: Verification successful at PETRONAS Seri Iskandar [4:15:47PM]

Pricing

Market Price Per Liter: RM 3.50

Subsidized Price Per Liter: RM 2.05

Transactions Log Page:



Transaction Log & Reports

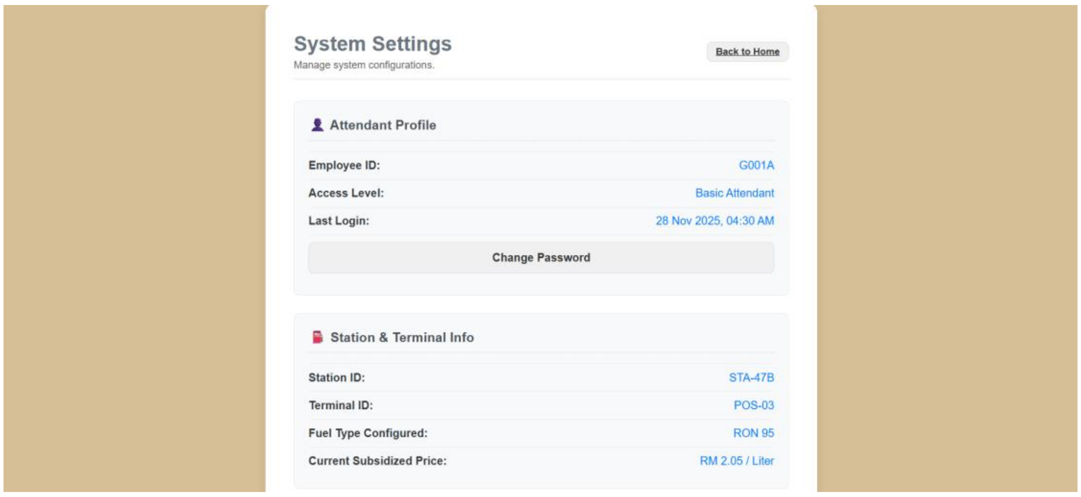
Review of transactions processed by Low Ren Jing.

Back to Home

Recent Transactions (Last 24 Hours)

TIME	REF ID	VEHICLE NO.	LITRES	PRICE/L	TOTAL (RM)	STATUS
14:35:12	T100192	BHH 4567	50.0	2.05	102.50	Subsidized
14:05:40	T100191	PGA 1122	40.0	3.50	140.00	Market Rate
13:45:01	T100190	WBB 7777	35.5	2.05	72.78	Subsidized
13:02:55	T100189	JEF 8080	60.0	2.05	123.00	Subsidized
12:30:10	T100188	SAD 4433	10.0	3.50	35.00	Market Rate
12:15:22	T100187	KJK 9911	55.0	2.05	112.75	Subsidized
11:45:30	T100186	LMN 5005	20.0	3.50	70.00	Market Rate
11:10:05	T100185	DDA 1001	45.0	2.05	92.25	Subsidized
10:40:48	T100184	MZA 4739	30.0	2.05	61.50	Subsidized
10:15:15	T100183	QWE 7771	15.0	3.50	52.50	Market Rate

Settings Page:



System Settings

Manage system configurations.

Back to Home

Attendant Profile

Employee ID:	G001A
Access Level:	Basic Attendant
Last Login:	28 Nov 2025, 04:30 AM

Change Password

Station & Terminal Info

Station ID:	STA-47B
Terminal ID:	POS-03
Fuel Type Configured:	RON 95
Current Subsidized Price:	RM 2.05 / Liter

<u>Design Rationale</u>

- Primary Rationale: To maximize transaction processing efficiency, clarity, and trustworthiness, directly supporting customers who requires speed during rush hour.
- The POS interface prioritizes simple, touch-friendly design to maximize speed and minimize cognitive load for the attendant during high-volume periods.
- The eligibility confirmation page provides instant, clear, and unambiguous visual feedback, a critical usability heuristic for a system handling finance.
- The success state is communicated powerfully through:
  - A large green checkmark icon.
  - Bold, high-contrast messaging ("ELIGIBLE - SUBSIDIZED PRICE").

  To ensure strong visual feedback.

2.3 USABILITY EVALUATION

2.3.1 Usability Test Plan

Test Scenario

The participant acts as Adam, the gas station attendant. Their task is to process three consecutive customers under typical morning rush conditions.

1. Customer A: Eligible for Subsidy
   - Scan QR → System displays "Eligible – Subsidized Price"
   - Attendant completes transaction
2. Customer B: Not Eligible
   - Scan QR → System displays "Not Eligible – Market Price"
   - Attendant charges normal price and ends transaction
3. Customer C: QR Scan Failure
   - Attempt scan → Error message appears
   - Participant must manually enter MyKad number and continue the process

Metrics Measured

- **Task Completion Time:** How long each transaction takes.
- **Error Rate:** Incorrect taps, retry loops, or misinterpretation of eligibility.
- **User Satisfaction:** Confidence level using the POS during high-pressure situations.
- **Learnability:** How quickly the participant adapts to the POS interface.

2.3.2 Design Improvements Based on Feedback

Based on the hypothetical usability test, two improvements were identified:

1. **Clearer Eligibility Status Indicators**
   - Issue: Participant hesitated when distinguishing between "Eligible" and "Not Eligible."
   - Improvement: Use colour-coded result cards (Green = Eligible, Red = Not Eligible) to speed up recognition.

2. **More Visible Recovery Actions for Scan Failures**
   - Issue: Participant struggled to locate the manual entry option after a failed scan.
   - Improvement: Add a large, high contrast "Re-scan / Manual Entry" button directly under the error message for quicker recovery.