POS Backend Full Documentation

1. Database Setup

We are using SQLite for offline/local POS.

1þ ã Install SQLite (or ensure better-sqlite3 is installed via npm):
```
npm install better-sqlite3
```

2þ ã Create the database file:
```
./database/offline_pos.db
```

3þ ã Example SQL Schema:

```
-- Users
CREATE TABLE users (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    name TEXT NOT NULL,
    email TEXT UNIQUE,
    phone TEXT,
    password_hash TEXT,
    role TEXT DEFAULT 'worker',
    pin TEXT,
    is_active BOOLEAN DEFAULT 1,
    created_at DATETIME DEFAULT CURRENT_TIMESTAMP
);

-- Products
CREATE TABLE products (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    name TEXT NOT NULL,
    price REAL NOT NULL,
    stock INTEGER DEFAULT 0,
    barcode TEXT,
    description TEXT,
    sku TEXT,
    created_at DATETIME DEFAULT CURRENT_TIMESTAMP
);

-- Product Images
CREATE TABLE product_images (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    product_id INTEGER NOT NULL,
    url TEXT,
    created_at DATETIME DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY(product_id) REFERENCES products(id)
);
```

```
-- Orders
CREATE TABLE orders (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    user_id INTEGER,
    total REAL,
    tax REAL DEFAULT 0,
    paid_amount REAL,
    change_amount REAL,
    payment_method TEXT,
    note TEXT,
    created_at DATETIME DEFAULT CURRENT_TIMESTAMP
);

-- Order Items
CREATE TABLE order_items (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    order_id INTEGER NOT NULL,
    product_id INTEGER,
    name TEXT,
    unit_price REAL,
    quantity INTEGER,
    total REAL,
    FOREIGN KEY(order_id) REFERENCES orders(id),
    FOREIGN KEY(product_id) REFERENCES products(id)
);

-- Audit Log
CREATE TABLE audit_logs (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    actor_id INTEGER,
    actor_role TEXT,
    action TEXT,
    details TEXT,
    created_at DATETIME DEFAULT CURRENT_TIMESTAMP
);
```

2. Authentication

Run createadmin.js to create the test admin user.

```
    POST /api/auth/login
Body (JSON):
{
  "email": "admin@example.com",
  "password": "yourpassword"
}
Response:
{
  "token": "JWT_TOKEN_HERE",
  "user": { "id": 1, "name": "Admin", "role": "admin" }
```

}
Use this token in Authorization header for protected routes:
Authorization: Bearer <token>

3. Users (Workers)

POST /api/users/workers (Admin only)
Body (JSON):
```
{
  "name": "John Doe",
  "email": "john@example.com",
  "phone": "12345678",
  "password": "changeme",
  "pin": "1234"
}
```
Response: User object

GET /api/users/workers (Admin only)
Response: List of worker objects

4. Products

GET /api/products
Optional query: /api/products?q=searchterm
Response: Array of products

GET /api/products/:id
Response: Product object including images

POST /api/products
Body (JSON):
```
{
  "name": "Sample Product",
  "price": 12.5,
  "stock": 10,
  "barcode": "123456789",
  "description": "Test product",
  "sku": "SP-001"
}
```
Response: Product object

PUT /api/products/:id
Body (JSON): Fields to update
```
{
  "price": 15,
  "stock": 20
}
```

5. Orders

POST /api/orders

Body (JSON):
```
{
  "items": [
    { "product_id": 1, "quantity": 2 },
    { "product_id": 2, "quantity": 1 }
  ],
  "paid_amount": 50,
  "payment_method": "cash",
  "note": "First order"
}
```
Response: Order object + receipt

GET /api/orders
Response: List of latest 200 orders

6. Reports

GET /api/reports/daily
Response:
```
[
  { "day": "2025-10-30", "orders_count": 5, "total_sales": 123.5 },
  ...
]
```