AutoParts Marketplace Platform – Developer SRS

1. Introduction

1.1 Purpose

Defines functional and non-functional requirements of a B2B multi-vendor marketplace for new automotive parts, enabling vendors to sell via approved storefronts, and including full admin and super admin control.

1.2 Scope

Supports vendor registration, employee role management, product management with admin approval, customer browsing and ordering, and real-time messaging. Admin and Superadmin oversee platform-wide management, including marketplace toggling and analytics.

1.3 Definitions

- **Super Admin:** Has all powers of Admin but is hidden from Admin views.
- **Admin:** Platform manager with broad permissions.
- **Vendor Owner, Vendor Employee:** Manage stores and orders with defined roles.
- **Customer:** Product shopper and buyer.
- **SKU:** Stock Keeping Unit.
- **Condition:** Product state; defaulted as 'new'.

2. Overall Description

2.1 Product Perspective

Multi-vendor marketplace implemented with Node, React, Next.js, and PostgreSQL.

2.2 Product Functions

- Vendor registration requiring admin manual approval.
- Static employee roles (predefined, shown in dropdowns) assigned to vendor employees.
- Product addition with required admin approval before public display.

- CSV import for new products only; rejects on any invalid record.
- Barcode/QR SKU scanner feature planned.
- Real-time chat support maintaining conversation history per logged-in user (continuing chats).
- Admin and superadmin have full monitoring rights including chat logs.
- Commission per sale with selectable/custom percentages.
- Multilingual support with language management by superadmin.
- Admin-specific marketplace sales with show/hide toggle.

2.3 User Classes

- **Super Admin:** Complete access equal to Admin; actions not visible to Admin users.
- **Admin:** Manages vendors, products, orders, returns, discounts, and platform analytics.
- **Vendor Owner:** Manages store, employees, products (post-approval), and orders.
- **Vendor Employee:** Assigned static roles from predefined set.
- **Customer:** Browses, orders, and chats.

3. Specific Requirements

3.1 User Roles and Permissions

- Employee roles are static; stored to populate dropdowns.
- All critical actions by superadmin, admin, or employee are **logged with user IDs** for audit.

3.2 Vendor Registration and Commission

- Vendor registration is manual approval via admin panel.
- Commission rates selectable or custom, stored in settings/configurable table.

3.3 Product Management

- Products include Name, Description, SKU, Price, Stock, Images, Make/Model/Year, and 'condition' defaulting to 'new'.
- Barcode and QR code SKU scanning supported for product lookup and entry.
- Vendor products require admin approval before public listing.

- Vehicles and models marked inactive remain for historical data integrity.

3.4 Import & Image Processing

- CSV import **only for adding** products; strict validation rejecting entire files on errors.
- Images uploaded with cropping interface; saved as processed files; DB saves references, not raw files.

3.5 Search and Filtering

- Filtering on vehicle compatibility, category, brand, condition, and price.
- No fuzzy search.
- Hidden/suspended vendors and inactive products do not appear in search.

3.6 Orders & Payments

- Supports AED and INR currencies.
- Payment gateways include Stripe, PayPal, PayTabs, and CCAvenue.
- Tax, shipping, and discount are aggregated and shown as single amounts in cart/checkout.
- Instant order confirmation on payment success.
- Admin handles all return/refund processes.

3.7 Messaging System

- Real-time chat maintaining history per logged-in session.
- Admin and superadmin monitor chats and can flag content.
- Chats link to specific orders.

3.8 Discounts and Promotions

- Discounts/coupons can be global or vendor-restricted at creation.
- Admin sets categories/products for vendor-specific discounts.
- Promotional banners support manual and scheduled activations.

3.9 Analytics and Reports

Admin Analytics:

- Total sales (platform-wide)
- New vendor registrations
- Vendor activity and approval status
- Sales by vendor, category, product
- Orders by status (pending, shipped, canceled)
- Revenue breakdown by currency
- Customer behavior metrics (visits, conversion rates)
- Return/refund statistics

Vendor Analytics:

- Store sales overview
- Top-selling products
- Recent order statuses
- Customer messages and response rates
- Inventory alerts

3.10 Admin Marketplace

- Admin may sell products on a dedicated marketplace section.
- E-commerce toggle option to show/hide admin marketplace storefront without removing data.

4. Non-Functional Requirements

4.1 Performance

- Key pages (Homepage, Product, Category) load under **3 seconds** on broadband.
- System supports **100 concurrent active users** performing browsing and checkout actions without slowdowns.

4.2 Security

- Passwords hashed and salted.

- Mandatory HTTPS.
- Protection from common web attacks (SQL injection, XSS).

4.3 Usability

- Mobile-first responsive UI.
- Consistent and intuitive navigation.

4.4 Reliability

- 99.5% uptime excluding scheduled maintenance.
- Regular automated backup.

5. CRUD-Level Operations per Module

5.1 Admin Module

- **Products**
  - Create own products.
  - Read own and vendor products.
  - Update own products.
  - Delete own products (soft delete).
  - Approve or disable vendor products.
- **Vendors**
  - Create vendor accounts.
  - Read vendor data.
  - Update vendor details.
  - Disable or soft delete vendor accounts.
- **Expenses**
  - Create expense records.
  - Read expense reports.
  - Update expense entries.

    - Soft delete expenses.
- **Employees**
    - Create employee accounts.
    - Read employee profiles.
    - Update employee records.
    - Soft delete employee accounts.
- **Orders**
    - Read all orders.
    - Update order status.
    - Soft delete order records (archival).
- **Discounts and Coupons**
    - Create discount campaigns.
    - Read active and past discounts.
    - Update discount parameters.
    - Soft delete obsolete discounts.
- **Promotional Banners**
    - Create banners.
    - Read and list banners.
    - Update banner settings.
    - Soft delete banners.
- **Languages**
    - Add/Import languages.
    - Read language settings.
    - Update translations.
    - Soft delete language entries.

5.2 Super Admin Module

- Inherits all Admin CRUD operations.

- Perform CRUD on Admin users (create, read, update, disable, soft delete).
- Manage system-wide configurations and settings.
- Hidden from Admin view but logs all actions.

5.3 Vendor Module

- **Products**
    - Create own products (pending admin approval).
    - Read own products.
    - Update own product details.
    - Soft delete own products.
- **Employees**
    - Read employees assigned to own store.
    - No direct creation or deletion—Admin or Vendor owner manages.
- **Orders**
    - Read orders belonging to own store.
    - Update order status if permitted.
- **Messages**
    - Read and send messages related to owned orders.
- **Reports and Analytics**
    - Read own sales and product analytics.

5.4 Vendor Employee Module

- Limited based on assigned static roles.
- **Products**
    - Read own store products as per permission.
    - Update or create products only if role permits.
- **Orders**
    - Read and process orders assigned.

- **Messages**
    - Send and receive messages within permitted scope.

5.5 Customer Module

- **Account**
    - Create and update own profile.
- **Orders**
    - Create new orders.
    - Read own order history.
- **Messages**
    - Initiate and continue messaging with vendors.

Notes:

- **Soft Deletes:** All the delete actions mark records as inactive or deleted without physical removal, enabling recovery and audit.
- **Audit Logs:** Every CRUD action by any user (Super Admin, Admin, Vendor, Employee) is logged with user ID, timestamp, and action details.
- **Role-Based Filtering:** Users see and operate only records permitted by their role and ownership

```
-- ENUM types
CREATE TYPE user_role AS ENUM ('superadmin', 'admin', 'vendor_owner', 'vendor_employee',
'customer');
CREATE TYPE employee_role AS ENUM ('manager', 'cashier', 'stock_manager', 'other');
CREATE TYPE order_status AS ENUM ('pending', 'confirmed', 'shipped', 'delivered',
'cancelled', 'returned');
CREATE TYPE product_condition AS ENUM ('new', 'used', 'refurbished');

-- USERS
CREATE TABLE users (
    id SERIAL PRIMARY KEY,
    username VARCHAR(50) UNIQUE NOT NULL,
    email VARCHAR(255) UNIQUE NOT NULL,
    password_hash VARCHAR(255) NOT NULL,
    role user_role NOT NULL,
    is_active BOOLEAN DEFAULT TRUE,
    created_at TIMESTAMP WITH TIME ZONE DEFAULT now(),
    updated_at TIMESTAMP WITH TIME ZONE DEFAULT now()
);

-- VENDORS
CREATE TABLE vendors (
    id SERIAL PRIMARY KEY,
    user_id INT REFERENCES users(id) ON DELETE CASCADE,
    is_active BOOLEAN DEFAULT TRUE,
    is_approved BOOLEAN DEFAULT FALSE,
    created_at TIMESTAMP WITH TIME ZONE DEFAULT now(),
    updated_at TIMESTAMP WITH TIME ZONE DEFAULT now()
);

-- STORES (One vendor can have multiple stores)
CREATE TABLE stores (
    id SERIAL PRIMARY KEY,
    vendor_id INT REFERENCES vendors(id) ON DELETE CASCADE,
    store_name VARCHAR(255) NOT NULL,
    is_active BOOLEAN DEFAULT TRUE,
    created_at TIMESTAMP WITH TIME ZONE DEFAULT now(),
    updated_at TIMESTAMP WITH TIME ZONE DEFAULT now()
);

-- EMPLOYEES
CREATE TABLE employees (
    id SERIAL PRIMARY KEY,
    vendor_id INT REFERENCES vendors(id) ON DELETE CASCADE,
    user_id INT REFERENCES users(id) ON DELETE CASCADE,
    role employee_role NOT NULL,
    is_active BOOLEAN DEFAULT TRUE,
    created_at TIMESTAMP WITH TIME ZONE DEFAULT now(),
    updated_at TIMESTAMP WITH TIME ZONE DEFAULT now()
);

-- PRODUCT CATEGORIES
CREATE TABLE product_cat (
    id SERIAL PRIMARY KEY,
    name VARCHAR(100) NOT NULL UNIQUE,
    is_active BOOLEAN DEFAULT TRUE
);

CREATE TABLE product_cat_sub (
    id SERIAL PRIMARY KEY,
    category_id INT REFERENCES product_cat(id),
    name VARCHAR(100) NOT NULL,
    is_active BOOLEAN DEFAULT TRUE
);

-- VEHICLE MAKES, MODELS, YEARS
CREATE TABLE vehicle_makes (
    id SERIAL PRIMARY KEY,
    name VARCHAR(100) UNIQUE NOT NULL,
    is_active BOOLEAN DEFAULT TRUE
);

CREATE TABLE vehicle_models (
    id SERIAL PRIMARY KEY,
    make_id INT REFERENCES vehicle_makes(id),
    name VARCHAR(100) NOT NULL,
    is_active BOOLEAN DEFAULT TRUE
);

CREATE TABLE vehicle_years (
    id SERIAL PRIMARY KEY,
    year INT CHECK (year > 1900 AND year < 2100),
    is_active BOOLEAN DEFAULT TRUE
);

-- PRODUCTS
CREATE TABLE products (
    id SERIAL PRIMARY KEY,
    store_id INT REFERENCES stores(id) ON DELETE CASCADE,
    sku VARCHAR(100) UNIQUE NOT NULL,
    name VARCHAR(255) NOT NULL,
    description TEXT,
    price NUMERIC(10, 2) NOT NULL CHECK(price >= 0),
    stock INT NOT NULL CHECK(stock >= 0),
    condition product_condition DEFAULT 'new',
    product_cat_id INT REFERENCES product_cat(id),
    product_cat_sub_id INT REFERENCES product_cat_sub(id),
    is_active BOOLEAN DEFAULT FALSE, -- pending admin approval
    created_at TIMESTAMP WITH TIME ZONE DEFAULT now(),
    updated_at TIMESTAMP WITH TIME ZONE DEFAULT now()
);

-- PRODUCT IMAGES
CREATE TABLE product_images (
    id SERIAL PRIMARY KEY,
    product_id INT REFERENCES products(id) ON DELETE CASCADE,
    image_url TEXT NOT NULL,
    sort_order INT DEFAULT 0
);

-- PRODUCT VEHICLE COMPATIBILITY
CREATE TABLE product_vehicle_compatibility (
    id SERIAL PRIMARY KEY,
    product_id INT REFERENCES products(id) ON DELETE CASCADE,
    make_id INT REFERENCES vehicle_makes(id),
    model_id INT REFERENCES vehicle_models(id),
    year_id INT REFERENCES vehicle_years(id)
);

-- ORDERS
CREATE TABLE orders (
    id SERIAL PRIMARY KEY,
    customer_id INT REFERENCES users(id),
    total_amount NUMERIC(10,2) NOT NULL CHECK(total_amount >= 0),
    status order_status DEFAULT 'pending',
    created_at TIMESTAMP WITH TIME ZONE DEFAULT now(),
    updated_at TIMESTAMP WITH TIME ZONE DEFAULT now()
);

-- ORDER ITEMS
CREATE TABLE order_items (
    id SERIAL PRIMARY KEY,
    order_id INT REFERENCES orders(id) ON DELETE CASCADE,
    product_id INT REFERENCES products(id),
    quantity INT NOT NULL CHECK(quantity > 0)
);

-- MESSAGES
CREATE TABLE messages (
    id SERIAL PRIMARY KEY,
    order_id INT REFERENCES orders(id),
    sender_id INT REFERENCES users(id),
    receiver_id INT REFERENCES users(id),
    message TEXT NOT NULL,
    sent_at TIMESTAMP WITH TIME ZONE DEFAULT now(),
    is_read BOOLEAN DEFAULT FALSE
);

-- DISCOUNTS
CREATE TABLE discounts (
    id SERIAL PRIMARY KEY,
    code VARCHAR(50) UNIQUE NOT NULL,
    description TEXT,
    discount_percent NUMERIC(5, 2) CHECK(discount_percent > 0 AND discount_percent <=
100),
    valid_from DATE NOT NULL,
    valid_to DATE NOT NULL,
    vendor_id INT REFERENCES vendors(id) NULL,
    product_cat_id INT REFERENCES product_cat(id) NULL,
    product_id INT REFERENCES products(id) NULL,
    is_active BOOLEAN DEFAULT TRUE
);

-- PROMOTIONAL BANNERS
CREATE TABLE banners (
    id SERIAL PRIMARY KEY,
    title VARCHAR(255) NOT NULL,
    image_url TEXT,
    vendor_id INT REFERENCES vendors(id) NULL,
    product_cat_id INT REFERENCES product_cat(id) NULL,
    product_id INT REFERENCES products(id) NULL,
    is_active BOOLEAN DEFAULT FALSE,
    start_date DATE,
```

```sql
    end_date DATE
);

-- EXPENSES
CREATE TABLE expenses (
    id SERIAL PRIMARY KEY,
    vendor_id INT REFERENCES vendors(id) ON DELETE CASCADE,
    amount NUMERIC(15, 2) NOT NULL,
    description TEXT,
    expense_date DATE NOT NULL,
    created_at TIMESTAMP WITH TIME ZONE DEFAULT now()
);

-- LANGUAGES
CREATE TABLE languages (
    language_code VARCHAR(10) PRIMARY KEY,
    language_name VARCHAR(50) NOT NULL,
    is_active BOOLEAN DEFAULT TRUE
);

-- SETTINGS
CREATE TABLE settings (
    key VARCHAR(100) UNIQUE NOT NULL PRIMARY KEY,
    value TEXT NOT NULL,
    description TEXT NULL,
    created_at TIMESTAMP WITH TIME ZONE DEFAULT now(),
    updated_at TIMESTAMP WITH TIME ZONE DEFAULT now()
);

-- AUDIT LOGS
CREATE TABLE audit_logs (
    id SERIAL PRIMARY KEY,
    user_id INT REFERENCES users(id),
    action TEXT NOT NULL,
    detail JSONB,
    created_at TIMESTAMP WITH TIME ZONE DEFAULT now()
);

-- FUNCTIONS
-- Get products filtered by make, model, year
CREATE OR REPLACE FUNCTION get_filtered_products(make VARCHAR, model VARCHAR, year INT)
RETURNS TABLE(
    product_id INT,
    name VARCHAR,
    price NUMERIC(10,2),
    stock INT,
    condition product_condition
) AS $$
BEGIN
    RETURN QUERY
    SELECT p.id, p.name, p.price, p.stock, p.condition
    FROM products p
    JOIN product_vehicle_compatibility pvc ON pvc.product_id = p.id
    JOIN vehicle_makes vm ON pvc.make_id = vm.id
    JOIN vehicle_models vmo ON pvc.model_id = vmo.id
    JOIN vehicle_years vy ON pvc.year_id = vy.id
    WHERE vm.name = make AND vmo.name = model AND vy.year = year
      AND p.is_active = TRUE;
END;
$$ LANGUAGE plpgsql;

-- Log audit action
CREATE OR REPLACE FUNCTION log_action(uid INT, act TEXT, det JSONB DEFAULT '{}')
RETURNS VOID AS $$
BEGIN
    INSERT INTO audit_logs(user_id, action, detail) VALUES (uid, act, det);
END;
$$ LANGUAGE plpgsql;
```