# DELODUR POS System - Complete Documentation

## Table of Contents



---

## System Overview

### What is DELODUR POS System?

The DELODUR POS System is a comprehensive **Point of Sale (POS) and Inventory Management System** specifically designed for automotive parts retail operations. It provides a modern, touch-friendly interface for managing sales, inventory, suppliers, and business analytics.

### Key Capabilities

- **Real-time POS Operations** with tax calculation and receipt generation
- **Inventory Management** with barcode/QR code support
- **Supplier Management** and purchase order tracking
- **Multi-location Warehouse** management
- **AI-powered Chatbot** for customer support
- **Comprehensive Reporting** and analytics
- **User Authentication** with role-based access control

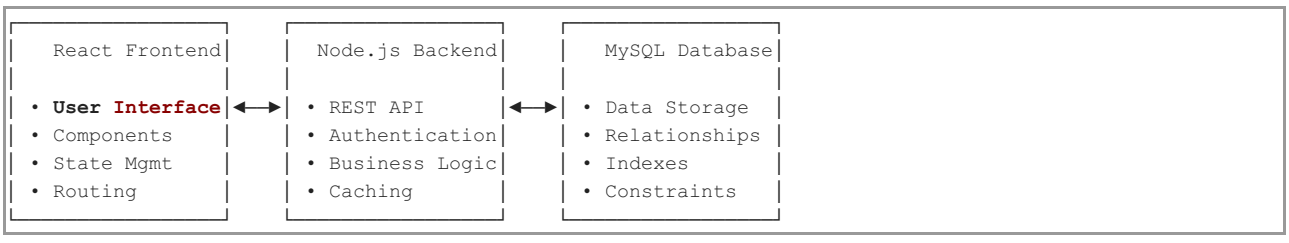
### Target Users

- **Retail Store Managers** - Oversee operations and view reports
- **Cashiers** - Process sales transactions
- **Warehouse Staff** - Manage inventory and stock movements
- **Administrators** - Configure system settings and manage users

---

## Architecture

### System Architecture Overview




### Frontend Architecture

- **React 18** with functional components and hooks
- **React Router** for client-side routing
- **React Bootstrap** for UI components
- **Chart.js** for data visualization
- **Custom CSS** with automotive-inspired design

### Backend Architecture

- **Express.js** web framework
- **RESTful API** design
- **JWT Authentication** for security
- **Connection Pooling** for database efficiency
- **Caching Layer** for performance optimization
- **Rate Limiting** for API protection

### Database Architecture

- **MySQL** relational database
- **Normalized Schema** with proper relationships
- **Indexed Fields** for query optimization
- **Foreign Key Constraints** for data integrity
- **Audit Trails** with timestamps

---

## ✨ Features

### 🛒 Point of Sale (POS)

- **Product Search** by name, barcode, or product code
- **Shopping Cart** with quantity controls
- **Tax Calculation** (12% Philippine tax)
- **Receipt Generation** with print support
- **Payment Processing** with multiple currency support
- **Transaction History** tracking

### 📦 Inventory Management

- **Real-time Stock Levels** monitoring
- **Barcode/QR Code** scanning support
- **Stock Movements** tracking (incoming/outgoing)
- **Reorder Point** alerts
- **Multi-location** warehouse support
- **Product Variations** (color, size, etc.)

### 👥 User Management

- **Role-based Access** (Admin, User)
- **JWT Authentication** with secure sessions
- **User Activity** logging
- **Password Security** with bcrypt hashing

### 📊 Reporting & Analytics

- **Sales Reports** with date filtering
- **Inventory Reports** with stock levels
- **Supplier Reports** with purchase history
- **Dashboard** with key metrics
- **Chart Visualizations** for trends

### 🤖 AI Chatbot

- **OpenAI Integration** for intelligent responses
- **Customer Support** automation
- **Product Information** assistance
- **Order Status** inquiries

### 🏢 Supplier Management

- **Supplier Database** with contact information
- **Purchase Order** tracking
- **Supplier Performance** analytics
- **Contact Management** with phone/email

---

## 🛠️ Technology Stack

### Frontend Technologies

| Technology | Version | Purpose |
|---|---|---|
| React | 18.2.0 | UI Framework |
| React Router | 6.20.1 | Navigation |
| React Bootstrap | 2.10.10 | UI Components |
| Chart.js | 4.5.0 | Data Visualization |
| Axios | 1.6.2 | HTTP Client |
| Bootstrap Icons | 1.11.6 | Icons |

### Backend Technologies

| Technology | Version | Purpose |
|---|---|---|
| Node.js | 18+ | Runtime Environment |
| Express.js | 4.18.2 | Web Framework |
| MySQL2 | 3.14.1 | Database Driver |
| JWT | 9.0.2 | Authentication |
| bcryptjs | 2.4.3 | Password Hashing |
| Helmet | 7.1.0 | Security Headers |

### Database & Storage

| Technology | Version | Purpose |
|---|---|---|
| MySQL | 8.0+ | Primary Database |
| NodeCache | 5.1.2 | In-Memory Caching |
| Redis | 4.6.12 | Session Storage |

### Development Tools

| Technology | Version | Purpose |
|---|---|---|
| Nodemon | 3.0.1 | Development Server |
| React Scripts | 5.0.1 | Build Tools |
| ESLint | - | Code Quality |

---

## 🚀 Installation & Setup

### Prerequisites

- **Node.js** (v16 or higher)
- **MySQL** (v8.0 or higher)
- **Git** for version control
- **npm** or **yarn** package manager

### Step 1: Clone Repository

```
git clone https://github.com/JanwelCast012010/delodur-pos-system.git
cd delodur-pos-system
```

### Step 2: Install Dependencies

```
# Install server dependencies
npm install

# Install client dependencies
cd client
npm install
cd ..
```

### Step 3: Database Setup

```
# Create MySQL database
mysql -u root -p
CREATE DATABASE inventory_system;
USE inventory_system;

# Import database schema
source database.sql;
```

### Step 4: Environment Configuration

```
# Copy environment template
cp env.example .env

# Edit .env file with your settings
DB_HOST=localhost
DB_USER=root
DB_PASSWORD=your_password
DB_NAME=inventory_system
JWT_SECRET=your_secure_jwt_secret
PORT=5000
```

### Step 5: Start Application

```
# Start backend server
npm start

# Start frontend (in new terminal)
cd client
npm start
```

### Step 6: Access Application

- **Frontend**: http://localhost:3000
- **Backend API**: http://localhost:5000
- **Default Login**: admin / password

---

## Database Schema

### Core Tables

#### Users Table

```
CREATE TABLE users (
  id INT AUTO_INCREMENT PRIMARY KEY,
  username VARCHAR(50) UNIQUE NOT NULL,
  password VARCHAR(255) NOT NULL,
  role ENUM('admin', 'user') DEFAULT 'user',
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

#### Products Table

```
CREATE TABLE products (
  id INT AUTO_INCREMENT PRIMARY KEY,
  barcode VARCHAR(50) UNIQUE,
  benz_number VARCHAR(20),
  brand VARCHAR(50) NOT NULL,
  alt_number VARCHAR(50),
  description TEXT,
  application TEXT,
  unit VARCHAR(20),
  reorder_point INT DEFAULT 0,
  location VARCHAR(50),
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP
);
```

#### Stock Items Table

```
CREATE TABLE stock_items (
  id INT AUTO_INCREMENT PRIMARY KEY,
  product_id INT NOT NULL,
  color_code VARCHAR(20),
  remarks VARCHAR(100),
  cost DECIMAL(10,2) DEFAULT 0,
  selling_price DECIMAL(10,2) DEFAULT 0,
  quantity INT DEFAULT 0,
  currency VARCHAR(10) DEFAULT 'USD',
  fc_cost DECIMAL(10,2) DEFAULT 0,
  conversion_rate DECIMAL(10,4) DEFAULT 1,
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  FOREIGN KEY (product_id) REFERENCES products(id) ON DELETE CASCADE
);
```

#### Sales History Table

```
CREATE TABLE sales_history (
  id INT AUTO_INCREMENT PRIMARY KEY,
  date DATE NOT NULL,
  receipt_number VARCHAR(50) NOT NULL,
  customer VARCHAR(100),
  product_id INT NOT NULL,
  stock_item_id INT NOT NULL,
  quantity INT NOT NULL,
  selling_price DECIMAL(10,2),
  total_amount DECIMAL(10,2),
  tax_amount DECIMAL(10,2),
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  FOREIGN KEY (product_id) REFERENCES products(id),
  FOREIGN KEY (stock_item_id) REFERENCES stock_items(id)
);
```

### Relationships

- **Products** → **Stock Items** (One-to-Many)
- **Products** → **Sales History** (One-to-Many)
- **Suppliers** → **Incoming Stocks** (One-to-Many)
- **Users** → **All Operations** (Authentication)

---

## API Documentation

### Authentication Endpoints

#### POST /api/auth/login

#### Login user and return JWT token

```
{
  "username": "admin",
  "password": "password"
}
```

#### Response:

```
{
  "token": "jwt_token_here",
  "user": {
    "id": 1,
    "username": "admin",
    "role": "admin"
  }
}
```

### Products Endpoints

#### GET /api/products

#### Get all products with pagination

```
Query Parameters:
- page: Page number (default: 1)
- limit: Items per page (default: 10)
- search: Search term
- brand: Filter by brand
```

#### POST /api/products

#### Create new product

```
{
  "barcode": "123456789",
  "brand": "Toyota",
  "description": "Oil Filter",
  "unit": "piece",
  "reorder_point": 10
}
```

### Sales Endpoints

#### POST /api/sales

#### Create new sale transaction

```
{
  "customer": "John Doe",
  "items": [
    {
      "product_id": 1,
      "stock_item_id": 1,
      "quantity": 2,
      "selling_price": 100.00
    }
  ]
}
```

### Inventory Endpoints

#### GET /api/stock

#### Get current stock levels

```
Query Parameters:
- product_id: Filter by product
- low_stock: Show only low stock items
```

#### POST /api/stock/incoming

#### Add incoming stock

```
{
  "date": "2024-01-15",
  "reference": "PO-001",
  "supplier_id": 1,
  "items": [
    {
      "product_id": 1,
      "stock_item_id": 1,
      "quantity": 50,
      "cost": 80.00
    }
  ]
}
```

---

# User Guide

## Getting Started

### 1. Login

- Navigate to http://localhost:3000
- Enter credentials: `admin` / `password`
- Click "Login" button

### 2. Dashboard Overview

The dashboard provides:

- **Total Products**: Number of products in catalog
- **Total Stock Items**: Available inventory items
- **Total Sales**: Today's sales count
- **Inventory Value**: Total value of current stock
- **Quick Actions**: Direct access to key functions

## Sales Operations

### Processing a Sale

1. **Navigate to Sales** section
2. **Search for Products**:
    - Type product name or code
    - Scan barcode/QR code
    - Browse product catalog
3. **Add to Cart**:
    - Click product to add
    - Adjust quantity with +/- buttons
    - Review cart items
4. **Complete Sale**:
    - Review totals and tax
    - Click "Complete Sale"
    - Print receipt (optional)

### Sales Features

- **Real-time Tax Calculation** (12% Philippine tax)
- **Multiple Currency Support** (USD, PHP)
- **Receipt Generation** with company branding
- **Transaction History** tracking

## Inventory Management

### Viewing Stock Levels

1. **Navigate to Stocks** section
2. **Filter Options**:
    - Search by product name/code
    - Filter by brand
    - Show low stock items only
3. **Stock Information**:
    - Current quantity
    - Cost and selling price
    - Location and remarks

### Adding New Products

1. **Navigate to Products** section

2. **Click "Add Product"**
3. **Fill Product Details**:
    - Basic information (name, brand, description)
    - Barcode/QR code
    - Unit and reorder point
    - Location information
4. **Save Product**

#### Managing Stock Items

1. **Select Product** from catalog
2. **Add Stock Item**:
    - Color/variation details
    - Cost and selling price
    - Initial quantity
    - Currency and conversion rates
3. **Save Stock Item**

### Reporting

#### Sales Reports

1. **Navigate to Reports** section
2. **Select Report Type**:
    - Daily sales summary
    - Monthly sales analysis
    - Product performance
3. **Set Date Range**
4. **Generate Report**

#### Inventory Reports

1. **Stock Level Report**:
    - Current quantities
    - Low stock alerts
    - Value calculations
2. **Movement Report**:
    - Incoming stock
    - Outgoing stock
    - Stock transfers

---

## 🔧 Administration Guide

### User Management

#### Creating New Users

1. **Access Admin Panel**
2. **Navigate to Users** section
3. **Click "Add User"**
4. **Fill User Details**:
    - Username (unique)
    - Password (secure)
    - Role (admin/user)
5. **Save User**

#### Managing User Roles

- **Admin**: Full system access

- **User**: Limited access to sales and inventory

### System Configuration

#### Database Backup

```
# Create backup
mysqldump -u root -p inventory_system > backup.sql

# Restore backup
mysql -u root -p inventory_system < backup.sql
```

#### Environment Variables

```
# Production settings
NODE_ENV=production
PORT=5000
DB_HOST=localhost
DB_USER=root
DB_PASSWORD=secure_password
DB_NAME=inventory_system
JWT_SECRET=very_secure_jwt_secret
```

### Performance Monitoring

#### Database Performance

- Monitor query execution times
- Check index usage
- Review connection pool status

#### Application Performance

- Monitor API response times
- Check memory usage
- Review cache hit rates

---

## Troubleshooting

### Common Issues

#### Database Connection Issues

**Problem**: Cannot connect to MySQL database
**Solution**:

1. Verify MySQL service is running
2. Check database credentials in .env
3. Ensure database exists
4. Test connection manually

#### Frontend Not Loading

**Problem**: React app won't start
**Solution**:

1. Check Node.js version (v16+)
2. Clear npm cache: `npm cache clean --force`
3. Delete node_modules and reinstall
4. Check for port conflicts

#### API Errors

**Problem**: Backend API returning errors
**Solution**:

1. Check server logs for errors
2. Verify database connection

3. Check JWT token validity
4. Review API endpoint URLs

### Error Codes

| Code | Description | Solution |
|---|---|---|
| 401 | Unauthorized | Check JWT token |
| 403 | Forbidden | Verify user permissions |
| 404 | Not Found | Check API endpoint |
| 500 | Server Error | Check server logs |

### Log Files

- **Application Logs**: Check console output
- **Database Logs**: MySQL error log
- **Access Logs**: Express.js access logs

---

## Performance Optimization

### Database Optimization

- **Indexed Fields**: Frequently queried columns
- **Connection Pooling**: Efficient database connections
- **Query Optimization**: Optimized SQL queries
- **Caching**: Redis for session storage

### Frontend Optimization

- **Code Splitting**: Lazy loading of components
- **Image Optimization**: Compressed images
- **Bundle Optimization**: Minified JavaScript
- **CDN Usage**: Static asset delivery

### Backend Optimization

- **Caching**: NodeCache for API responses
- **Rate Limiting**: API protection
- **Compression**: Gzip response compression
- **Security Headers**: Helmet middleware

---

## Security

### Authentication Security

- **JWT Tokens**: Secure session management
- **Password Hashing**: bcrypt for password security
- **Token Expiration**: Automatic token refresh
- **Role-based Access**: User permission control

### API Security

- **Rate Limiting**: Prevent API abuse
- **Input Validation**: Sanitize user inputs
- **SQL Injection Protection**: Parameterized queries
- **CORS Configuration**: Cross-origin protection

### Data Security

- **Encrypted Passwords**: bcrypt hashing
- **Secure Headers**: Helmet middleware
- **HTTPS**: Production SSL/TLS
- **Database Security**: User permissions

### Best Practices

1. **Regular Updates**: Keep dependencies updated
2. **Security Audits**: Regular vulnerability scans
3. **Backup Strategy**: Regular data backups
4. **Monitoring**: Log monitoring and alerts

---

## System Statistics

### Code Metrics

- **Total Lines**: 34,630 lines
- **Files**: 51 files
- **Languages**: 6 different technologies
- **Components**: 15 React components

### File Distribution

- **JavaScript**: 29 files (9,101 lines)
- **JSON**: 6 files (20,344 lines)
- **CSS**: 4 files (3,353 lines)
- **SQL**: 5 files (427 lines)
- **HTML**: 2 files (637 lines)
- **Markdown**: 5 files (768 lines)

### Database Tables

- **Users**: Authentication and user management
- **Products**: Product catalog and information
- **Stock Items**: Inventory with variations
- **Sales History**: Transaction records
- **Suppliers**: Vendor information
- **Warehouse**: Multi-location storage
- **Cashier**: POS session management

---

## Future Enhancements

### Planned Features

1. **Mobile App**: React Native mobile application
2. **Advanced Analytics**: Machine learning insights
3. **Multi-store Support**: Chain store management
4. **E-commerce Integration**: Online sales platform
5. **Advanced Reporting**: Custom report builder

### Technical Improvements

1. **Microservices Architecture**: Service decomposition
2. **Real-time Updates**: WebSocket integration
3. **Advanced Caching**: Redis cluster
4. **Load Balancing**: Multiple server instances
5. **Containerization**: Docker deployment

---

## Support & Contact

### Documentation

- **System Documentation**: This file
- **API Documentation**: REST API endpoints
- **User Manual**: Step-by-step guides
- **Troubleshooting**: Common issues and solutions

### Support Channels

- **GitHub Issues**: Bug reports and feature requests
- **Email Support**: Technical support inquiries

- **Documentation**: Self-service help resources

### Development Team

- **Lead Developer**: Janwel Cast
- **Repository**: https://github.com/JanwelCast012010/delodur-pos-system
- **Version**: 1.0.0

---



*This documentation is maintained by the development team and should be updated with each system release.*