Table of Contents

[1. Introduction 2](#_Toc181181902)

[2. Retailer Shop Context 2](#_Toc181181903)

[3. Invoice Context 3](#_Toc181181904)

[4. Application Requirements 4](#_Toc181181905)

[4.1 Use Case 4](#_Toc181181906)

[4.2 User Journey Flow 5](#_Toc181181907)

[4.3 Functional Requirements 5](#_Toc181181908)

[4.4 Technical Requirements 5](#_Toc181181909)

[4.5 Performance Requirements 5](#_Toc181181910)

[4.6 Security Requirements 6](#_Toc181181911)

[4.7 User Experience Requirements 6](#_Toc181181912)

[5. Functional Specifications 6](#_Toc181181913)

[5.1 System Overview 6](#_Toc181181914)

[5.2 User Roles and Permissions 6](#_Toc181181915)

[5.3 Functional Requirements 6](#_Toc181181916)

[5.4 User Interface Requirements 7](#_Toc181181917)

[5.5 System Requirements 7](#_Toc181181918)

[5.6 Non-Functional Requirements 7](#_Toc181181919)

[5.7 Acceptance Criteria 7](#_Toc181181920)

## 1. Introduction

This document combines the Application Requirements and Functional Specifications for the Retailer Application. The purpose is to capture user needs, business objectives, and system functionality for guiding the development process.

## 2. Retailer Shop Context

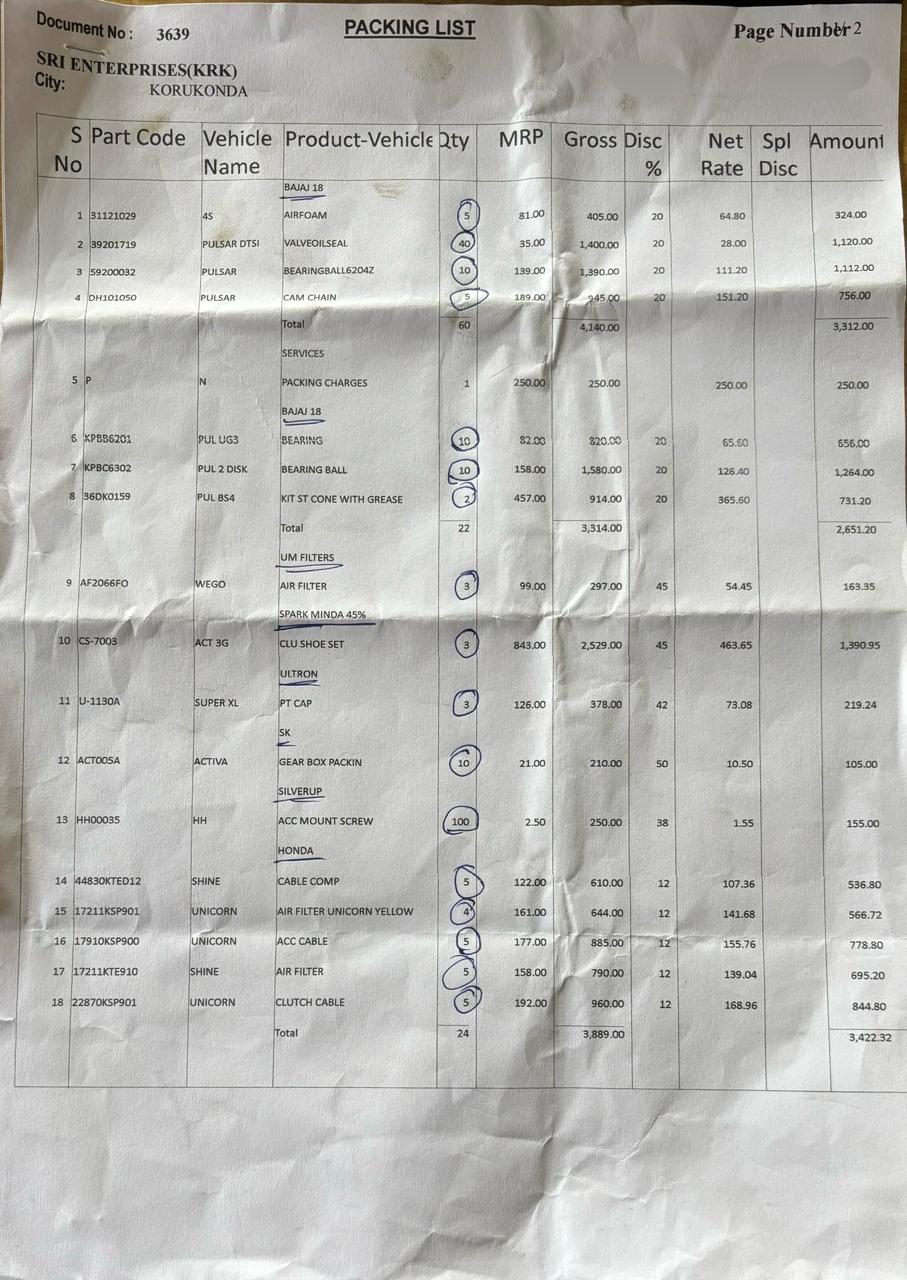


The retailer runs an **automobile spare parts shop** with a wide variety of items, including essential and specialized components for different vehicle types. Daily operations involve a hands-on approach to managing orders and inventory.

When new stock arrives, the retailer **sorts through items manually**, carefully checking each part against the invoice to ensure all ordered products are present. This process is time-consuming, requiring an item-by-item match, which can be challenging as it often interrupts customer interactions since most customers are **walk-ins**. After confirming that each item has arrived as ordered, the retailer **refills parts in designated areas** of the shop to keep everything organized and easily accessible.

This routine of verifying and stocking parts helps maintain accuracy in inventory but can occasionally disrupt the flow of business, especially during busy hours.

## 3. Invoice Context

****

The attached invoice provides a comprehensive breakdown of the products ordered by the retailer. The invoice includes the following columns:

* **S No**: Serial number for each line item.
* **Part Code**: Unique code assigned to each product part.
* **Vehicle Name**: The model or type of vehicle the part is associated with.
* **Product-Vehicle**: Specific product name alongside the compatible vehicle.
* **Qty**: Quantity of each item ordered, as per the retailer's requirements.
* **MRP**: Maximum Retail Price of each item.
* **Gross**: Gross discount applied on the MRP.
* **Disc %**: Discount percentage offered on each item.
* **Net Rate**: Rate after discount adjustments.
* **Spl Disc**: Special discount, if applicable.
* **Amount**: Final amount calculated for each item, post-discount.

In this particular invoice, the retailer has manually marked the **Qty** (Quantity) column to assist in verifying that each item has been received as per the order. By marking these quantities, the retailer can cross-check with the order he placed initially to ensure accuracy, a process that can be time-consuming and prone to discrepancies.

## 4. Application Requirements

### ****4.1 Use Case****

* **Scenario**: A retailer logs an order, photographs an invoice for OCR-based verification, and identifies missing items.
* **Steps**:
  1. Retailer saves order details (company name, product name, quantity).
  2. If necessary, edits or merges saved orders
  3. After order delivery, uploads an invoice for OCR-based verification.
  4. Compares saved order data with the OCR results to find discrepancies.

### ****4.2 User Journey Flow****

* **Order Entry** ➔ **Saving** ➔ **Order Editing or Merging (if necessary)** ➔ **Invoice Upload** ➔ **Discrepancy Identification**

### ****4.3 Functional Requirements****

* **Order Management**: Users save orders by entering details (company name, product name, quantity). A dropdown provides official names when a local name is typed.
* **Order Editing and Merging**: Retailers can edit or merge saved orders if necessary according to the requirement
* **Invoice Verification with OCR**: After saving an order, the retailer photographs the invoice, and OCR scans to read key fields (company name, product name, quantity) for comparison with saved orders.
* **Discrepancy Identification**: The retailer checks the OCR results against saved order data to find any discrepancies, such as missing items from the order.

.

### ****4.4 Technical Requirements****

* **Database**: Cloud-based, enabling real-time access and edits across devices.
* **OCR Technology**: Integrated OCR to capture invoice details for verification.
* **Technology Stack**: Supports both iOS and Android, with cloud and OCR processing.

### ****4.5 Performance Requirements****

* **Speed**: Fast OCR processing for a seamless experience.
* **Scalability**: Cloud setup should support high volumes of orders and invoices.

### ****4.6 Security Requirements****

* **Data Encryption**: Encrypt all data stored in the cloud.
* **User Authentication**: Secure login for managing sensitive data.

### ****4.7 User Experience Requirements****

* **Ease of Navigation**: Simplified order entry, invoice upload, and discrepancy review.
* **Dropdown Suggestions**: Official product names displayed based on local inputs.
* **Cloud Accessibility**: Data accessible from multiple devices.

## ****5. Functional Specifications****

### ****5.1 System Overview****

The application helps an automobile spare parts retailer log orders, manage order data through cloud storage, and verify invoices via OCR

### ****5.2 User Roles and Permissions****

* **Retailer**: Full access to log, verify, and edit orders.
* **Admin:** Manages app settings.

### ****5.3 Functional Requirements****

* **Order Logging**: Retailers log order details with dropdown suggestions showing official names for local terms.
* **Order Editing and Merging**: Retailers can adjust saved orders if necessary, based on requirement
* **OCR-Based Invoice Verification**: After receiving an order, retailers photograph an invoice, allowing OCR to capture details and check for discrepancies.
* **Discrepancy Review**: Retailers identify missing products by comparing invoice data against saved orders.

### ****5.4 User Interface Requirements****

* **Home Screen**: Access to view orders, add new ones, and upload invoices.
* **Order Details Screen**: Editable fields with dropdown suggestions for official names.
* **Invoice Upload Screen**: Allows image uploads for OCR, with extracted data shown for easy review.
* **Discrepancy Review Screen**: Displays saved orders alongside OCR data for identifying missing items.

### ****5.5 System Requirements****

* **Device Compatibility**: Android and iOS support.
* **Online Functionality**: Requires internet for cloud-based storage.
* **Cloud Storage**: Data stored centrally for easy access.

### ****5.6 Non-Functional Requirements****

* **Reliability**: High uptime for cloud storage.
* **Usability**: Intuitive interface.
* **Scalability**: High data volume support.

### ****5.7 Acceptance Criteria****

* **Order Logging**: Allows accurate logging and dropdown suggestions for official names.
* **Order Editing and Merging**: Allows modifications to saved orders based on discrepancies
* **OCR Processing and Verification**: Successfully scans invoices and identifies discrepancies.