Requirements Specification Document

Distribution Center Package Management System

Version: 1.0
Date: November 5, 2025
Project: Barcode-Based Package Management System
Prepared by: Development Team

Table of Contents

---

1. Introduction

1.1 Purpose

This document specifies the functional and non-functional requirements for the Distribution Center Package Management System, a barcode-based solution for tracking, categorizing, and storing packages in a distribution center.

1.2 Scope

The system will automate package registration through barcode scanning, automatically categorize packages based on their attributes, assign optimal storage locations, and provide comprehensive tracking and reporting capabilities.

1.3 Intended Audience

- Development Team
- Quality Assurance Team
- Distribution Center Operations Manager
- System Administrators
- Stakeholders

1.4 Definitions and Acronyms

- **DC**: Distribution Center
- **ACID**: Atomicity, Consistency, Isolation, Durability
- **SQLite**: Embedded relational database management system
- **NFR**: Non-Functional Requirement
- **FR**: Functional Requirement

---

2. System Overview

2.1 System Description

The Distribution Center Package Management System is a command-line application that processes packages received at a distribution center. The system scans barcodes, registers package details, automatically determines the appropriate category, and assigns storage locations based on predefined rules.

2.2 System Architecture

- **Frontend**: Command-line interface (CLI)
- **Backend**: Python application
- **Database**: SQLite
- **Deployment**: Single-user desktop application

---

3. Functional Requirements

FR1: Package Registration

ID: FR1
Priority: High
Description: The system shall scan and register package barcodes with complete package information.

Detailed Requirements

FR1.1 - The system shall accept barcode input through manual entry or automatic generation.

FR1.2 - The system shall capture the following package attributes:

- Barcode (unique identifier, 12 digits)
- Weight (in kilograms, positive decimal value)
- Dimensions: Length, Width, Height (in centimeters, positive decimal values)
- Destination (text string, minimum 3 characters)
- Priority level (Standard or Express)

FR1.3 - The system shall validate that barcode identifiers are unique in the database.

FR1.4 - The system shall reject registration if a duplicate barcode is detected.

FR1.5 - The system shall automatically generate a unique internal package ID upon successful registration.

FR1.6 - The system shall record the timestamp of package receipt (received_at).

FR1.7 - The system shall set the initial package status to "Received" upon registration.

FR1.8 - The system shall provide immediate feedback to the user upon successful or failed registration.

Acceptance Criteria

AC1.1 - Given a unique barcode is entered, when the package registration is submitted with valid data, then the package shall be stored in the database with all attributes preserved.

AC1.2 - Given a duplicate barcode is entered, when the package registration is submitted, then the system shall display an error message "Barcode [barcode] already exists in the system" and shall not create a duplicate record.

AC1.3 - Given invalid data is entered (negative weight, empty destination), when the package registration is submitted, then the system shall display a descriptive error message and shall not create a record.

AC1.4 - Given a package is successfully registered, when the operation completes, then the system shall display confirmation including the barcode, assigned category, and assigned location.

AC1.5 - Given multiple packages are registered sequentially, when each registration completes, then each package shall have a unique package_id generated by the database.

AC1.6 - Given a package is registered at any time, when the registration completes, then the received_at timestamp shall reflect the current system time with accuracy to the second.

AC1.7 - Given no barcode is provided by the user, when the user chooses to generate a barcode, then the system shall create a unique 12-digit numeric barcode automatically.

AC1.8 - Given a package registration is initiated, when the process completes within 2 seconds, then the performance requirement NFR1 is satisfied.

FR2: Category Management

ID: FR2
Priority: High
Description: The system shall support multiple package categories and automatically classify packages based on predefined rules.

Detailed Requirements

FR2.1 - The system shall support five predefined categories:

- Standard: Regular packages with standard delivery
- Express: High-priority expedited delivery
- Fragile: Delicate items requiring careful handling
- Heavy: Items exceeding weight thresholds
- International: International shipments

FR2.2 - The system shall automatically assign categories based on the following rules:

- Priority = "Express" → Express category
- Destination contains "International" OR multiple country separators → International category
- Weight > 50.0 kg → Heavy category
- Weight < 5.0 kg → Fragile category
- All other cases → Standard category

FR2.3 - The system shall store category information including:

- Category name
- Description
- Associated zone
- Maximum recommended weight
- Priority level

FR2.4 - The system shall allow retrieval of all packages within a specific category.

FR2.5 - Each category shall have a unique zone designation (A, B, C, D, E).

Acceptance Criteria

AC2.1 - Given a package with priority "Express", when categorization occurs, then the package shall be assigned to the Express category (category_id = 2).

AC2.2 - Given a package with weight 60 kg and priority "Standard", when categorization occurs, then the package shall be assigned to the Heavy category (category_id = 4).

AC2.3 - Given a package with weight 3 kg and priority "Standard", when categorization occurs, then the package shall be assigned to the Fragile category (category_id = 3).

AC2.4 - Given a package with destination "London, UK, International" and priority "Standard", when categorization occurs, then the package shall be assigned to the International category (category_id = 5).

AC2.5 - Given a package with weight 25 kg and priority "Standard" to a domestic destination, when categorization occurs, then the package shall be assigned to the Standard category (category_id = 1).

AC2.6 - Given multiple categorization rules could apply, when categorization occurs, then the system shall apply rules in the defined priority order (Express → International → Heavy → Fragile → Standard).

AC2.7 - Given a query for packages by category, when executed, then the system shall return all packages matching that category_id.

FR3: Location Assignment

ID: FR3
Priority: High
Description: The system shall automatically assign storage locations based on package category and location availability.

Detailed Requirements

FR3.1 - The system shall maintain a location inventory with the following structure:

- Location code (format: ZONE##-##, e.g., "A01-03")
- Zone (A, B, C, D, E)
- Aisle number (01-05)
- Shelf number (01-04)
- Associated category
- Occupancy status (occupied/available)

FR3.2 - The system shall create 20 locations per category zone (5 aisles × 4 shelves).

FR3.3 - The system shall automatically select the first available location within the appropriate category zone.

FR3.4 - The system shall mark a location as occupied immediately upon assignment to a package.

FR3.5 - The system shall prevent assignment of multiple packages to the same location.

FR3.6 - The system shall reject package registration if no available locations exist for the assigned category.

FR3.7 - The system shall support querying locations by zone, occupancy status, and category.

FR3.8 - The system shall release (mark as available) a location when the assigned package status changes to "Delivered".

Acceptance Criteria

AC3.1 - Given a package is categorized as Standard (zone A), when location assignment occurs, then the package shall be assigned a location with code matching pattern "A##-##".

AC3.2 - Given all locations in a category zone are occupied, when a new package of that category is registered, then the system shall display an error message "No available locations for category [category_name]" and shall not complete the registration.

AC3.3 - Given two packages are registered simultaneously for the same category, when location assignment occurs, then each package shall receive a different location_id.

AC3.4 - Given a location is assigned to a package, when the assignment completes, then the location's is_occupied flag shall be set to TRUE (1).

AC3.5 - Given a package status is updated to "Delivered", when the status update completes, then the associated location's is_occupied flag shall be set to FALSE (0).

AC3.6 - Given a query for available locations in zone B, when executed, then the system shall return only locations where zone = 'B' AND is_occupied = 0.

AC3.7 - Given the database is initialized, when location creation completes, then there shall be exactly 100 total locations (5 zones × 20 locations each).

FR4: Package Tracking

ID: FR4
Priority: High
Description: The system shall track package status throughout the distribution lifecycle and maintain a complete audit trail.

Detailed Requirements

FR4.1 - The system shall support the following package statuses:

- Received: Package has been scanned but not yet stored
- Stored: Package has been assigned a location
- In Transit: Package is being moved or shipped
- Delivered: Package has reached final destination

FR4.2 - The system shall allow status updates for any package via barcode lookup.

FR4.3 - The system shall record every status change in an audit trail with:

- Package ID
- Action type
- Old status
- New status
- Old location (if applicable)
- New location (if applicable)
- Timestamp
- Notes

FR4.4 - The system shall allow searching packages by:

- Exact barcode match

- Category
- Current location
- Current status

FR4.5 - The system shall display complete package information including:

- All package attributes
- Current category assignment
- Current location
- Current status
- Registration timestamp

FR4.6 - The system shall maintain historical audit records indefinitely (no automatic deletion).

FR4.7 - The system shall prevent deletion of package records that have audit trail entries.

Acceptance Criteria

AC4.1 - Given a valid barcode is provided, when a package search is performed, then the system shall return all current package details within 1 second.

AC4.2 - Given a package status is updated, when the update completes, then a new audit trail record shall be created with old_status, new_status, and current timestamp.

AC4.3 - Given a package does not exist, when searched by barcode, then the system shall display "Package with barcode [barcode] not found".

AC4.4 - Given a package status changes from "Stored" to "In Transit", when the update occurs, then the audit trail shall show action = "STATUS_UPDATE", old_status = "Stored", new_status = "In Transit".

AC4.5 - Given multiple status changes occur for a package, when audit history is queried, then all changes shall be retrievable in chronological order.

AC4.6 - Given a package is delivered, when the status is set to "Delivered", then the package location shall be freed and marked available for new packages.

AC4.7 - Given a package has been registered, when attempting to delete the package record directly, then the database foreign key constraint shall prevent deletion if audit records exist.

FR5: Reporting

ID: FR5
Priority: Medium
Description: The system shall generate comprehensive reports on package distribution, location utilization, and system activities.

Detailed Requirements

FR5.1 - The system shall generate a summary report containing:

- Package count by category
- Package count by status
- Location occupancy by zone
- Recent activity log (last 10 actions)

FR5.2 - The system shall calculate location occupancy percentage per zone using the formula:

Occupancy % = (Occupied Locations / Total Locations) × 100

FR5.3 - The system shall display recent activities in reverse chronological order (most recent first).

FR5.4 - The system shall format reports in a user-friendly tabular structure.

FR5.5 - Report generation shall complete within 2 seconds regardless of database size (up to 10,000 packages).

FR5.6 - The system shall display zero counts for categories with no packages.

Acceptance Criteria

AC5.1 - Given packages exist in the system, when a summary report is requested, then the report shall display count of packages for each of the five categories.

AC5.2 - Given zone A has 12 occupied locations out of 20, when the report calculates occupancy, then it shall display "60.0% occupied".

AC5.3 - Given 15 audit actions have occurred, when the recent activities section is displayed, then it shall show exactly the 10 most recent actions.

AC5.4 - Given no packages exist in the Fragile category, when the report is generated, then the report shall show "Fragile: 0 packages".

AC5.5 - Given the database contains 10,000 package records, when a report is requested, then the report shall be generated and displayed within 2 seconds.

AC5.6 - Given audit trail records exist, when recent activities are displayed, then each entry shall include timestamp, barcode, action type, and notes.

---

4. Non-Functional Requirements

NFR1: Performance

ID: NFR1
Priority: High
Description: The system shall meet specified performance benchmarks for all operations.

Detailed Requirements

NFR1.1 - Barcode scanning and package registration shall complete within 2 seconds from submission to confirmation display.

NFR1.2 - Database queries for package search shall return results within 1 second.

NFR1.3 - Report generation shall complete within 2 seconds for databases containing up to 10,000 package records.

NFR1.4 - The system shall support concurrent registration of up to 50 packages within a 1-minute window.

NFR1.5 - Location assignment queries shall complete within 500 milliseconds.

Acceptance Criteria

AC-NFR1.1 - Given a package registration is submitted, when measured from submit to confirmation, then 95% of operations shall complete within 2 seconds under normal load.

AC-NFR1.2 - Given a barcode search is executed, when measured from query initiation to result display, then the operation shall complete within 1 second.

AC-NFR1.3 - Given a database with 10,000 records exists, when a summary report is generated, then the operation shall complete within 2 seconds.

AC-NFR1.4 - Given 50 packages are registered within 60 seconds, when all operations complete, then no transactions shall be lost or corrupted.

NFR2: Reliability

ID: NFR2
Priority: High
Description: The system shall maintain data integrity and operate reliably under normal and error conditions.

Detailed Requirements

NFR2.1 - All database operations shall follow ACID transaction principles (Atomicity, Consistency, Isolation, Durability).

NFR2.2 - The system shall use database foreign key constraints to maintain referential integrity.

NFR2.3 - The system shall implement rollback mechanisms for failed transactions.

NFR2.4 - The system shall validate all user inputs before database operations.

NFR2.5 - The system shall handle database connection failures gracefully with appropriate error messages.

NFR2.6 - The system shall prevent data corruption through proper exception handling.

NFR2.7 - The system shall maintain a 99.9% uptime during operational hours.

Acceptance Criteria

AC-NFR2.1 - Given a database transaction fails midway, when the failure occurs, then all changes within that transaction shall be rolled back.

AC-NFR2.2 - Given an attempt to delete a category with associated packages, when the delete operation is attempted, then the foreign key constraint shall prevent the deletion.

AC-NFR2.3 - Given invalid data is submitted (e.g., negative weight), when validation occurs, then the system shall reject the data before any database operation.

AC-NFR2.4 - Given a database connection error occurs, when the error is encountered, then the system shall display a user-friendly error message without crashing.

AC-NFR2.5 - Given the system operates for 24 hours, when uptime is calculated, then availability shall be at least 99.9% (maximum 86 seconds downtime).

NFR3: Usability

ID: NFR3
Priority: Medium
Description: The system shall provide an intuitive and user-friendly interface for distribution center staff.

Detailed Requirements

NFR3.1 - The command-line interface shall present a clear numbered menu with all available options.

NFR3.2 - Error messages shall be descriptive and provide actionable guidance to the user.

NFR3.3 - Success confirmations shall include relevant details (barcode, category, location).

NFR3.4 - The system shall use visual indicators (✅, ❌, 📦, 📊) to enhance readability.

NFR3.5 - Menu options shall be accessible via single-digit numeric input.

NFR3.6 - The system shall provide "Press Enter to continue" prompts to control screen flow.

NFR3.7 - All prompts shall clearly indicate required input format and acceptable values.

Acceptance Criteria

AC-NFR3.1 - Given the main menu is displayed, when a user views the screen, then all six menu options shall be clearly numbered and described.

AC-NFR3.2 - Given an invalid barcode is entered, when the error is displayed, then the message shall explain why the barcode is invalid and what format is expected.

AC-NFR3.3 - Given a package is successfully registered, when confirmation is shown, then it shall display barcode, category name, and location code.

AC-NFR3.4 - Given any operation completes, when the result is displayed, then appropriate visual indicators (✅ for success, ❌ for error) shall be shown.

AC-NFR3.5 - Given a user is presented with a menu choice, when the user enters a single digit, then the corresponding action shall be executed.

NFR4: Maintainability

ID: NFR4
Priority: Medium
Description: The system shall be designed for easy maintenance, updates, and debugging.

Detailed Requirements

NFR4.1 - Code shall follow PEP 8 Python style guidelines.

NFR4.2 - All functions and classes shall include docstrings explaining purpose, parameters, and return values.

NFR4.3 - Database schema shall be normalized to at least Third Normal Form (3NF).

NFR4.4 - The system shall use parameterized SQL queries to prevent SQL injection.

NFR4.5 - Code shall be organized into logical classes with single responsibilities.

NFR4.6 - Magic numbers shall be avoided; constants shall be named and documented.

NFR4.7 - The system shall include inline comments for complex logic.

Acceptance Criteria

AC-NFR4.1 - Given the Python code is analyzed, when checked against PEP 8, then no violations shall be present.

AC-NFR4.2 - Given any function or class is reviewed, when examined, then it shall have a complete docstring.

AC-NFR4.3 - Given the database schema is analyzed, when checked for normalization, then it shall meet 3NF requirements (no transitive dependencies).

AC-NFR4.4 - Given any SQL query in the code, when reviewed, then it shall use parameterized placeholders (?) rather than string concatenation.

AC-NFR4.5 - Given the code structure is reviewed, when analyzed, then each class shall have a single, well-defined responsibility.

NFR5: Scalability

ID: NFR5
Priority: Low
Description: The system shall support future growth in data volume and feature expansion.

Detailed Requirements

NFR5.1 - The system shall support storage of at least 10,000 package records without performance degradation.

NFR5.2 - Database tables shall use indexes on frequently queried columns (barcode, category_id, location_id).

NFR5.3 - The database schema shall allow addition of new categories without code changes.

NFR5.4 - The location structure shall support expansion to additional zones beyond A-E.

NFR5.5 - The system shall use auto-incrementing primary keys to support unlimited record growth.

Acceptance Criteria

AC-NFR5.1 - Given the database contains 10,000 package records, when a barcode search is performed, then results shall be returned within 1 second.

AC-NFR5.2 - Given database tables are analyzed, when examining the schema, then indexes shall exist on packages.barcode, packages.category_id, and packages.location_id.

AC-NFR5.3 - Given a new category is added to the Categories table, when packages are registered, then the new category shall be available without application code changes.

AC-NFR5.4 - Given package, category, and location records are created, when examining primary keys, then all shall use auto-incrementing integer values.

NFR6: Security

ID: NFR6
Priority: Medium
Description: The system shall protect data integrity and prevent common security vulnerabilities.

Detailed Requirements

NFR6.1 - The system shall use parameterized queries for all database operations.

NFR6.2 - The system shall validate and sanitize all user inputs.

NFR6.3 - The system shall enable foreign key constraints in SQLite.

NFR6.4 - The system shall prevent SQL injection attacks through proper query construction.

NFR6.5 - Database file permissions shall be set to restrict access to authorized users only.

Acceptance Criteria

AC-NFR6.1 - Given any SQL query in the codebase, when reviewed, then no query shall use string formatting or concatenation with user input.

AC-NFR6.2 - Given user input is received, when processed, then it shall be validated against expected types and ranges before use.

AC-NFR6.3 - Given the database connection is established, when checked, then "PRAGMA foreign_keys = ON" shall be executed.

AC-NFR6.4 - Given malicious input is attempted (e.g., "1' OR '1'='1"), when processed, then it shall be safely handled without causing SQL errors or unauthorized data access.

---

5. Test Cases for FR1: Package Registration

Test Case 1.1: Successful Package Registration with Valid Data

Test Case ID: TC-FR1-001
Priority: High
Requirement: FR1.1, FR1.2, FR1.5, FR1.6, FR1.7
Objective: Verify that a package with valid, complete data is successfully registered.

Preconditions:

- System is running and database is initialized
- Barcode "123456789012" does not exist in the database
- At least one location is available in the Standard category

Test Data:

- Barcode: 123456789012

- Weight: 15.5 kg
- Length: 30 cm
- Width: 20 cm
- Height: 15 cm
- Destination: "New York, USA"
- Priority: "Standard"

Test Steps:

1. Launch the application
2. Select option "1. Register New Package"
3. Enter barcode: 123456789012
4. Enter weight: 15.5
5. Enter length: 30
6. Enter width: 20
7. Enter height: 15
8. Enter destination: "New York, USA"
9. Enter priority: "Standard"

Expected Results:

- System displays: "✅ Package registered successfully!"
- System displays: "Barcode: 123456789012"
- System displays: "Category: Standard"
- System displays: "Location: A##-##" (where ## are valid numbers)
- Database verification:
    - New record exists in Packages table with barcode "123456789012"
    - package_id is auto-generated and unique
    - received_at timestamp is within 1 second of current time
    - status = "Stored"
    - category_id = 1 (Standard)
    - All entered data matches database values

Actual Results: *[To be filled during testing]*

Status: *[Pass/Fail]*

Notes: *[Any observations]*

Test Case 1.2: Duplicate Barcode Detection

Test Case ID: TC-FR1-002
Priority: High
Requirement: FR1.3, FR1.4
Objective: Verify that the system rejects registration when a duplicate barcode is detected.

Preconditions:

- System is running and database is initialized
- Package with barcode "999888777666" already exists in database

Test Data:

- Barcode: 999888777666 (duplicate)
- Weight: 10.0 kg

- Length: 25 cm
- Width: 15 cm
- Height: 10 cm
- Destination: "Chicago, USA"
- Priority: "Standard"

Test Steps:

1. Launch the application
2. Select option "1. Register New Package"
3. Enter barcode: 999888777666
4. Enter weight: 10.0
5. Enter length: 25
6. Enter width: 15
7. Enter height: 10
8. Enter destination: "Chicago, USA"
9. Enter priority: "Standard"

Expected Results:

- System displays: "❌ Error: Barcode 999888777666 already exists in the system!"
- No new record is created in Packages table
- Original record with same barcode remains unchanged
- Database transaction is rolled back (if applicable)
- Package count in database remains unchanged

Actual Results: *[To be filled during testing]*

Status: *[Pass/Fail]*

Notes: *[Any observations]*

Test Case 1.3: Invalid Weight (Negative Value)

Test Case ID: TC-FR1-003
Priority: High
Requirement: FR1.2, FR1.8
Objective: Verify that the system rejects registration with invalid weight value.

Preconditions:

- System is running and database is initialized

Test Data:

- Barcode: 111222333444
- Weight: -5.0 kg (invalid)
- Length: 20 cm
- Width: 15 cm
- Height: 10 cm
- Destination: "Boston, USA"
- Priority: "Standard"

Test Steps:

1. Launch the application
2. Select option "1. Register New Package"
3. Enter barcode: 111222333444
4. Enter weight: -5.0
5. Attempt to continue with remaining fields

Expected Results:

- System displays: "❌ Invalid input! Please enter valid numbers." or similar error message
- No record is created in Packages table
- User is returned to menu or given option to retry
- Database remains unchanged

Actual Results: *[To be filled during testing]*

Status: *[Pass/Fail]*

Notes: *[This test validates input validation at the UI level]*

Test Case 1.4: Empty Destination Field

Test Case ID: TC-FR1-004
Priority: Medium
Requirement: FR1.2, FR1.8
Objective: Verify that the system handles empty required fields appropriately.

Preconditions:

- System is running and database is initialized

Test Data:

- Barcode: 555666777888
- Weight: 12.0 kg
- Length: 25 cm
- Width: 20 cm
- Height: 15 cm
- Destination: "" (empty string)
- Priority: "Standard"

Test Steps:

1. Launch the application
2. Select option "1. Register New Package"
3. Enter barcode: 555666777888
4. Enter weight: 12.0
5. Enter length: 25
6. Enter width: 20
7. Enter height: 15
8. Press Enter without entering destination
9. Enter priority: "Standard"

Expected Results:

- System either:

    - Prompts again for destination with error message, OR
    - Rejects registration with error message about required fields
- No record is created in Packages table
- User is able to correct the input

Actual Results: *[To be filled during testing]*

Status: *[Pass/Fail]*

Notes: *[Current implementation may accept empty string; enhancement needed]*

Test Case 1.5: Automatic Barcode Generation

Test Case ID: TC-FR1-005
Priority: Medium
Requirement: FR1.1
Objective: Verify that the system can automatically generate a unique barcode when none is provided.

Preconditions:

- System is running and database is initialized

Test Data:

- Barcode: [Press Enter without input]
- Weight: 8.0 kg
- Length: 20 cm
- Width: 15 cm
- Height: 12 cm
- Destination: "Miami, USA"
- Priority: "Express"

Test Steps:

1. Launch the application
2. Select option "1. Register New Package"
3. Press Enter without entering a barcode
4. Note the generated barcode displayed
5. Enter weight: 8.0
6. Enter length: 20
7. Enter width: 15
8. Enter height: 12
9. Enter destination: "Miami, USA"
10. Enter priority: "Express"

Expected Results:

- System displays: "Generated barcode: [12-digit number]"
- Generated barcode is 12 digits
- Generated barcode is numeric only
- Package is registered successfully with generated barcode
- Database record contains the generated barcode
- Generated barcode is unique (not existing in database)

Actual Results: *[To be filled during testing]*

Status: *[Pass/Fail]*

Notes: *[Document generated barcode for verification]*

Test Case 1.6: Registration with Express Priority

Test Case ID: TC-FR1-006
Priority: High
Requirement: FR1.2, FR1.7, FR2.2
Objective: Verify that packages with Express priority are correctly registered and categorized.

Preconditions:

- System is running and database is initialized
- At least one location is available in Express category

Test Data:

- Barcode: 777888999000
- Weight: 5.0 kg
- Length: 15 cm
- Width: 12 cm
- Height: 10 cm
- Destination: "Seattle, USA"
- Priority: "Express"

Test Steps:

1. Launch the application
2. Select option "1. Register New Package"
3. Enter barcode: 777888999000
4. Enter weight: 5.0
5. Enter length: 15
6. Enter width: 12
7. Enter height: 10
8. Enter destination: "Seattle, USA"
9. Enter priority: "Express"

Expected Results:

- System displays: "✅ Package registered successfully!"
- System displays: "Category: Express"
- System displays: "Location: B##-##" (Zone B for Express)
- Database verification:
    - Record exists with barcode "777888999000"
    - category_id = 2 (Express)
    - priority = "Express"
    - location_id references a location in zone B
    - status = "Stored"

Actual Results: *[To be filled during testing]*

Status: *[Pass/Fail]*

Notes: *[Verifies integration with categorization logic]*

Test Case 1.7: Registration Timestamp Accuracy

Test Case ID: TC-FR1-007
Priority: Medium
Requirement: FR1.6
Objective: Verify that the received_at timestamp accurately reflects the time of registration.

Preconditions:

- System is running and database is initialized
- System clock is synchronized

Test Data:

- Barcode: 333444555666
- Weight: 10.0 kg
- Length: 20 cm
- Width: 15 cm
- Height: 10 cm
- Destination: "Portland, USA"
- Priority: "Standard"

Test Steps:

1. Note the current system time (to the second)
2. Launch the application
3. Select option "1. Register New Package"
4. Enter barcode: 333444555666
5. Complete registration with test data
6. Immediately query database for the received_at timestamp
7. Compare timestamp with noted system time

Expected Results:

- Database received_at timestamp is within 5 seconds of noted system time
- Timestamp format is valid SQLite datetime format (YYYY-MM-DD HH:MM:SS)
- Timestamp reflects the time of registration, not future or past dates

Actual Results: *[To be filled during testing]*

Status: *[Pass/Fail]*

Notes: *[Record both times for comparison]*

Test Case 1.8: Registration Performance

Test Case ID: TC-FR1-008
Priority: High
Requirement: FR1.8, NFR1.1
Objective: Verify that package registration completes within the 2-second performance requirement.

Preconditions:

- System is running and database is initialized

- Database contains fewer than 10,000 records
- System resources are normal (no high CPU/memory usage)

Test Data:

- Barcode: 444555666777
- Weight: 12.5 kg
- Length: 28 cm
- Width: 18 cm
- Height: 14 cm
- Destination: "Denver, USA"
- Priority: "Standard"

Test Steps:

1. Launch the application
2. Select option "1. Register New Package"
3. Start timer
4. Enter all package data
5. Submit registration
6. Stop timer when confirmation message appears
7. Record elapsed time

Expected Results:

- Total time from submission to confirmation is ≤ 2 seconds
- Registration completes successfully
- All data is correctly stored in database
- Performance meets NFR1.1 requirement

Actual Results: *[To be filled during testing]*

Elapsed Time: *[Record in seconds]*

Status: *[Pass/Fail]*

Notes: *[Note any performance issues or delays]*

Test Case 1.9: Registration with No Available Locations

Test Case ID: TC-FR1-009
Priority: High
Requirement: FR1.8, FR3.6
Objective: Verify that registration fails gracefully when no locations are available for the package category.

Preconditions:

- System is running and database is initialized
- All 20 locations in Standard category (Zone A) are occupied

Test Data:

- Barcode: 666777888999
- Weight: 15.0 kg
- Length: 30 cm
- Width: 20 cm

- Height: 15 cm
- Destination: "Austin, USA"
- Priority: "Standard"

Test Steps:

1. Verify all Zone A locations are occupied (run SQL query)
2. Launch the application
3. Select option "1. Register New Package"
4. Enter test data for a Standard category package
5. Attempt to complete registration

Expected Results:

- System displays: "❌ Error: No available locations for category Standard"
- No record is created in Packages table
- No location occupancy changes
- User is returned to main menu
- Database remains in consistent state

Actual Results: *[To be filled during testing]*

Status: *[Pass/Fail]*

Notes: *[This tests the integration between registration and location assignment]*

Test Case 1.10: Sequential Registration of Multiple Packages

Test Case ID: TC-FR1-010
Priority: Medium
Requirement: FR1.5, NFR1.4
Objective: Verify that multiple packages can be registered sequentially with unique IDs assigned.

Preconditions:

- System is running and database is initialized
- Sufficient locations are available

Test Data:

- Package 1: Barcode 111000111000, Weight 10 kg, Standard priority
- Package 2: Barcode 222000222000, Weight 20 kg, Standard priority
- Package 3: Barcode 333000333000, Weight 30 kg, Standard priority

Test Steps:

1. Launch the application
2. Register Package 1 with test data
3. Note the package_id from database
4. Register Package 2 with test data
5. Note the package_id from database
6. Register Package 3 with test data
7. Note the package_id from database
8. Query database for all three packages

Expected Results:

- All three packages are successfully registered
- Each package receives a unique, auto-incrementing package_id
- Package 1 ID < Package 2 ID < Package 3 ID
- Each package is assigned a different location
- All registrations complete without errors
- Database maintains referential integrity

Actual Results: *[To be filled during testing]*

Package IDs: *[Record: P1=*, **P2=**, *P3=__]_*

Status: *[Pass/Fail]*

Notes: *[Verify ID sequence is correct]*

Test Case 1.11: Registration with Boundary Weight Values

Test Case ID: TC-FR1-011
Priority: Medium
Requirement: FR1.2, FR2.2
Objective: Verify correct handling of packages with weight values at category boundaries.

Preconditions:

- System is running and database is initialized

Test Data: Test Set A - Fragile/Standard boundary:

- Barcode: 999111999111
- Weight: 5.0 kg (exactly at boundary)
- Priority: "Standard"

Test Set B - Standard/Heavy boundary:

- Barcode: 888222888222
- Weight: 50.0 kg (exactly at boundary)
- Priority: "Standard"

Test Steps:

1. Register package with weight 5.0 kg
2. Verify assigned category
3. Register package with weight 50.0 kg
4. Verify assigned category

Expected Results: Test Set A:

- Weight 5.0 kg should be categorized as Standard (rule: weight < 5.0 for Fragile)
- Location should be in Zone A

Test Set B:

- Weight 50.0 kg should be categorized as Standard (rule: weight > 50.0 for Heavy)
- Location should be in Zone A

Actual Results: *[To be filled during testing]*

Status: *[Pass/Fail]*

Notes: *[Boundary conditions are critical for correct categorization]*

Test Case 1.12: Registration with Special Characters in Destination

Test Case ID: TC-FR1-012
Priority: Low
Requirement: FR1.2
Objective: Verify that destination fields with special characters are handled correctly.

Preconditions:

- System is running and database is initialized

Test Data:

- Barcode: 555111555111
- Weight: 8.0 kg
- Length: 20 cm
- Width: 15 cm
- Height: 12 cm
- Destination: "São Paulo, Brazil - Rua O'Connor #45"
- Priority: "Standard"

Test Steps:

1. Launch the application
2. Select option "1. Register New Package"
3. Enter barcode: 555111555111
4. Enter test data including special characters in destination
5. Complete registration
6. Search for package by barcode
7. Verify destination is displayed correctly

Expected Results:

- Package is registered successfully
- Special characters (ã, ', #) are stored correctly in database
- Package search displays destination exactly as entered
- No character encoding issues occur
- Database maintains data integrity

Actual Results: *[To be filled during testing]*

Status: *[Pass/Fail]*

Notes: *[Test UTF-8 encoding support]*

Test Case 1.13: Registration Rollback on Failure

Test Case ID: TC-FR1-013
Priority: High
Requirement: FR1.8, NFR2.1, NFR2.3
Objective: Verify that database transaction is rolled back if registration fails midway.

Preconditions:

- System is running and database is initialized
- Ability to simulate database constraint violation

Test Data:

- Barcode: 123456789999
- Weight: 10.0 kg
- Other valid data

Test Steps:

1. Note current package count in database
2. Attempt registration that will fail (e.g., violate a constraint)
3. Verify database state after failure
4. Check package count in database
5. Verify no partial records exist

Expected Results:

- Registration fails with appropriate error message
- Database package count remains unchanged
- No orphaned records in Packages table
- No location is marked as occupied
- No audit trail entry is created
- Database maintains referential integrity

Actual Results: *[To be filled during testing]*

Status: *[Pass/Fail]*

Notes: *[Verifies ACID transaction principles]*

Test Case 1.14: Audit Trail Creation on Registration

Test Case ID: TC-FR1-014
Priority: High
Requirement: FR1.7, FR4.3
Objective: Verify that an audit trail entry is created when a package is registered.

Preconditions:

- System is running and database is initialized

Test Data:

- Barcode: 147258369147

- Weight: 11.0 kg
- Length: 25 cm
- Width: 18 cm
- Height: 13 cm
- Destination: "Phoenix, USA"
- Priority: "Standard"

Test Steps:

1. Launch the application
2. Register package with test data
3. Note the assigned location from confirmation message
4. Query AuditTrail table for entries with the new package_id
5. Verify audit trail content

Expected Results:

- One audit trail record exists for the new package
- Audit record contains:
  - action = "REGISTERED"
  - new_status = "Stored"
  - new_location = [assigned location code]
  - timestamp is accurate
  - notes contain category information
- Audit trail package_id matches the new package's package_id

Actual Results: *[To be filled during testing]*

Status: *[Pass/Fail]*

Notes: *[Verifies integration between registration and audit system]*

Test Case 1.15: Case-Insensitive Priority Handling

Test Case ID: TC-FR1-015
Priority: Low
Requirement: FR1.2, FR2.2
Objective: Verify that priority field accepts various case formats.

Preconditions:

- System is running and database is initialized

Test Data: Test variations:

- "express" (lowercase)
- "EXPRESS" (uppercase)
- "Express" (mixed case)
- "STANDARD" (uppercase)

Test Steps:

1. Register package with priority "express" (lowercase)
2. Verify category assignment
3. Register package with priority "EXPRESS" (uppercase)
4. Verify category assignment

5. Test similar variations for "Standard"

Expected Results:

- All case variations are accepted
- "express" (any case) results in Express category
- "standard" (any case) results in appropriate category based on other attributes
- No case-sensitivity errors occur
- Database stores the original case entered by user

Actual Results: *[To be filled during testing]*

Status: *[Pass/Fail]*

Notes: *[Documents current system behavior for case handling]*

---

6. Glossary

Audit Trail: A chronological record of all system activities and changes made to package records.

Barcode: A unique 12-digit numeric identifier assigned to each package for tracking purposes.

Category: A classification assigned to packages based on their attributes (Standard, Express, Fragile, Heavy, International).

Distribution Center (DC): A warehouse facility where packages are received, sorted, stored, and dispatched for delivery.

Location: A specific storage position within the distribution center, identified by zone, aisle, and shelf numbers.

Occupancy: The state of a storage location, indicating whether it is currently holding a package (occupied) or available for new packages.

Package: A physical item received at the distribution center for storage and eventual delivery.

Priority: The urgency level assigned to a package (Standard or Express).

Status: The current state of a package in the distribution workflow (Received, Stored, In Transit, Delivered).

Zone: A designated area within the distribution center assigned to a specific category (A, B, C, D, E).

Appendix A: Traceability Matrix

Test Case ID	Requirements Covered	Priority
TC-FR1-001	FR1.1, FR1.2, FR1.5, FR1.6, FR1.7	High
TC-FR1-002	FR1.3, FR1.4	High
TC-FR1-003	FR1.2, FR1.8	High
TC-FR1-004	FR1.2, FR1.8	Medium
TC-FR1-005	FR1.1	Medium
TC-FR1-006	FR1.2, FR1.7, FR2.2	High
TC-FR1-007	FR1.6	Medium
TC-FR1-008	FR1.8, NFR1.1	High
TC-FR1-009	FR1.8, FR3.6	High
TC-FR1-010	FR1.5, NFR1.4	Medium
TC-FR1-011	FR1.2, FR2.2	Medium
TC-FR1-012	FR1.2	Low
TC-FR1-013	FR1.8, NFR2.1, NFR2.3	High
TC-FR1-014	FR1.7, FR4.3	High
TC-FR1-015	FR1.2, FR2.2	Low

Appendix B: Test Data Management

Database Preparation Scripts

Reset Database for Testing:

sql

```sql
DELETE FROM AuditTrail;
DELETE FROM Packages;
UPDATE Locations SET is_occupied = 0;
DELETE FROM sqlite_sequence WHERE name IN ('Packages', 'AuditTrail');
```

Populate Test Barcode:

sql

```sql
INSERT INTO Packages (barcode, weight, length, width, height, destination, priority, category_id, location_id, status)
VALUES ('999888777666', 10.0, 25.0, 15.0, 10.0, 'Test City, USA', 'Standard', 1, 1, 'Stored');
```

Occupy All Zone A Locations (for TC-FR1-009):

sql

```
UPDATE Locations SET is_occupied = 1 WHERE zone = 'A';
```

Appendix C: Sign-Off

Role	Name	Signature	Date
Project Manager	_____	_____	_____
Lead Developer	_____	_____	_____
QA Lead	_____	_____	_____
Business Analyst	_____	_____	_____

Document Revision History

Version	Date	Author	Changes
1.0	2025-11-05	Development Team	Initial release

END OF DOCUMENT