**Axomate – Invoice & Customer/Vehicle Flow Requirements**

v1.0

*Author: Preet Chahal*

Date: August 22, 2025

# Index

# Overview

Scope: Desktop WPF app (Axomate) – customer/vehicle management, mileage capture, invoice creation, PDF generation.

Goal: Provide a clear, testable specification for the revised flows: Add/Save Customer; Add/Save Vehicle (with Mileage History); New Invoice → Save & Print (duplicate-safe); and the 5‑hour mileage lock.

# 1. Terminology

* SelectedCustomer: Currently selected existing customer from the list.
* NewCustomer: Editable buffer for creating or editing a customer from the form fields.
* SelectedVehicle: Currently selected saved vehicle for the selected customer.
* Vehicle: Editable buffer for creating or editing a vehicle from the form fields.
* Mileage / MileageText: Mileage bound to UI; MileageText is the raw string input, Mileage is parsed int? used by logic.
* NewCustomerMode / NewVehicleMode / NewInvoiceMode: UI mode flags that gate Save buttons.
* ServiceMomentNow: Timestamp composed from ServiceDate.Date + current time-of-day.

# 2. Functional Requirements

* FR‑1: Add New Customer
* 1) User clicks Add New Customer. System clears SelectedCustomer and VehiclesByCustomer; initializes blank NewCustomer; resets SelectedVehicle, Vehicle, Mileage, MileageText, and service history; sets NewCustomerMode = true and NewVehicleMode = false.
* 2) Save New Customer button is enabled only when NewCustomerMode == true.
* 3) On Save New Customer: validate Name (required); trim Name/Phone/Email/AddressLine1; insert via \_customerService.AddOrUpdateAsync with Id=0; refresh lookups and select saved customer; reflect values into NewCustomer; load empty vehicles; set NewCustomerMode = false.
* FR‑2: Add New Vehicle
* 1) Precondition: SelectedCustomer exists (Id > 0).
* 2) User clicks Add New Vehicle. If no customer, show message. Clear SelectedVehicle; init blank Vehicle tied to SelectedCustomer.Id; reset Mileage and MileageText; clear service history; set NewVehicleMode = true.
* 3) Save New Vehicle is enabled only when NewVehicleMode == true, SelectedCustomer.Id > 0, and Vehicle form is valid (VIN or License Plate or Make+Model).
* 4) On Save New Vehicle: trim fields; insert via \_vehicleService.AddOrUpdateAsync with Id=0; refresh vehicles; select saved; record mileage per FR‑4 using MileageText→ParseMileage(); set NewVehicleMode = false.
* FR‑3: New Invoice → Save & Print (duplicate-safe)
* 1) Save & Print disabled by default.
* 2) User clicks New Invoice: clear line items and totals; set ServiceDate = Now; clear Mileage and MileageText; set NewInvoiceMode = true.
* 3) Save & Print enabled only when NewInvoiceMode == true and IsBusy == false.
* 4) On Save & Print: set IsBusy = true; validate line items (at least one, Price>0, Quantity>0); map custom items to catalog; if customer/vehicle not saved, create them using FR‑1/FR‑2 validations; record mileage per FR‑4 at ServiceMomentNow; persist invoice via \_invoiceService.AddAsync with Mileage = parsed miles; set NewInvoiceMode = false after successful save; generate PDF with same mileage and timestamp; set IsBusy = false.
* 5) On failure, show error and keep NewInvoiceMode = true to allow retry.
* FR‑4: Mileage Capture (5‑hour rule)
* 1) On typing (MileageText changed): if value parses to m>0, set Mileage=m and call TryRecordMileageChangeAsync(m); if blank, Mileage=null.
* 2) On prefill (vehicle change or date change): fetch latest mileage for the day; set Mileage and MileageText under suppression (no recording).
* 3) 5‑hour lock applies to manual edits and save-time insert: if last record < 5h, block and revert to latest; show remaining lock time; Save-time adds only if no record in last 5h.
* 4) No regressions: if new value < latest persisted, ignore.

# 3. Validation Rules

* Customer: Name required; optional Phone/Email/Address trimmed.
* Vehicle (create): must have VIN or License Plate or (Make and Model); fields trimmed.
* Invoice Line Items: at least one; Price > 0; Quantity ≥ 1.
* Mileage: positive integer; not less than latest persisted; 5‑hour rule applies.

# 4. UI/UX Requirements

* Buttons and enablement:
* - Add New Customer: always enabled.
* - Save New Customer: enabled only when NewCustomerMode == true.
* - Add New Vehicle: enabled when a customer is selected or saved (SelectedCustomer.Id > 0).
* - Save New Vehicle: enabled only when NewVehicleMode == true and vehicle form valid.
* - New Invoice: always enabled.
* - Save & Print: enabled only when NewInvoiceMode == true and IsBusy == false.
* Mileage TextBox binds to MileageText (TwoWay, UpdateSourceTrigger=PropertyChanged).
* On customer change: clear vehicle selection, mileage fields, and service history.
* On vehicle change: prefill mileage for the day and load service history.

# 5. Persistence & Data Model

* CreatedDate / CreatedBy: all persisted tables include CreatedDate (UTC) and CreatedBy (string). Set on insert by services/repository.
* Tables: Customer, Vehicle, ServiceItem, Invoice, InvoiceLineItem, MileageHistory, Company, AdminCredential (as applicable).
* MileageHistory: VehicleId, Mileage, RecordedDate (timestamp), Source, Notes. Guard: 0 ≤ Mileage ≤ 2,000,000.
* Invoice: consider unique index (CustomerId, VehicleId, ServiceDate) to prevent duplicates.

# 6. Error Handling & Logging

* All command handlers catch exceptions, log to error.log, and show concise user messages.
* On mileage lock violation, show message with last recorded timestamp and remaining lock time.
* On unique constraint failures (e.g., duplicate license plate), show friendly message.

# 7. Non‑Functional Requirements

* Consistency: The same mileage value is used for DB and PDF.
* Responsiveness: Long operations set IsBusy and disable relevant commands.
* Stability: Suppression flags prevent recursive recordings during programmatic changes.

# 8. Acceptance Criteria (AC)

* AC‑1 Customer Save: After Add New Customer, entering a valid Name and saving creates and selects the customer; Save button disables.
* AC‑2 Vehicle Save with Mileage: After Add New Vehicle, entering valid fields and optional mileage saves vehicle; records mileage if last record ≥5h.
* AC‑3 New Invoice gates Save & Print: Save & Print disabled by default; enabled after New Invoice; disabled again after success.
* AC‑4 Mileage lock on typing: If last record <5h, revert to latest persisted and show lock message.
* AC‑5 Save‑time mileage insert: On Save & Print, inserts mileage only if last record ≥5h; otherwise none.
* AC‑6 Duplicate prevention: Save & Print disabled while busy; NewInvoiceMode=false after success.

# 9. Test Cases (high level)

* Create customer only; verify persistence and UI modes.
* Create vehicle only (validations fail/pass; unique plate conflict).
* Mileage input: invalid text, empty, decreasing, increasing, <5h blocked, ≥5h allowed.
* New Invoice → Save & Print: with new customer/vehicle; with existing; with custom items; with empty items (should fail).
* PDF contains the same mileage and timestamp as DB.
* Reset form clears modes/fields; Save & Print disabled.

# 10. Implementation Notes (traceability)

* Commands: AddNewCustomer/SaveNewCustomer (FR‑1); AddNewVehicle/SaveNewVehicle (FR‑2); CreateNewInvoice/SaveAndPrintInvoice (FR‑3); LoadServiceHistory.
* Mode flags: NewCustomerMode, NewVehicleMode, NewInvoiceMode.
* Mileage logic: OnMileageTextChanged, OnMileageChanged, PrefillMileageFromHistoryAsync, TryRecordMileageChangeAsync, RecordMileageOnSaveIfStaleAsync.
* Logging: LogErrorToFile.

# 11. Out of Scope (this iteration)

* Advanced customer/vehicle validation (email/phone formats, VIN checksum).
* Editing existing customer/vehicle during Save & Print path (handled by dedicated Save buttons).
* Multi‑currency or tax rules beyond existing fields.

# 12. Open Questions

* What value populates CreatedBy (Windows user, app login, or static)?
* Should Save & Print update an Invoice Status (e.g., Draft → Final) or is creation implicit?
* Do we allow orphan vehicles (no customer)? Currently No.

Note: In Word, press F9 or use References → Update Table to refresh the Index (TOC).