# CareLite - Functional Specification Document

## 1) Project Summary

CareLite is a lightweight web application for a small outpatient clinic. Staff register people, schedule time‑bound appointments with providers, document visits, generate bills, record payments, view basic reports, and (later) send notifications on key events. The system is delivered in phases and deployed after each phase.

**Note:** This specification intentionally does **not** define entities or schema. Please propose and refine the domain model during analysis/refinement.

## 2) User Roles (minimum)

* **Admin:** Can manage staff/providers, view and manage billing, access reports, configure system settings.
* **Staff:** Performs day‑to‑day operations (registration, scheduling, updates, payments).
* **Clinician:** Records visit notes and views relevant schedules.

You may refine, add, or merge roles as needed.

## 3) Functional Modules & Requirements

### 3.1 Authentication & Session

* Users can sign in with credentials and receive a session token.
* Access to protected features requires being signed in.
* Users can sign out; expired sessions are rejected gracefully.
* (Later) Users can refresh sessions without re‑entering credentials.

### 3.2 Registration & Record Management

* Staff can create a new **core record** (e.g., a person or patient) with identity and contact details.
* Staff can search and list existing records using text and simple filters.
* The system helps avoid duplicates (e.g., by flagging potential matches).
* All create/update/delete actions produce a human‑readable audit entry.

### 3.3 Scheduling

* Staff can create, edit, and cancel **appointments** between a person and a provider.
* Appointments have a start time and duration chosen from allowed options (e.g., 15/30/60 minutes).
* Appointments must fall within configured business hours.
* The system prevents double‑booking of the same provider/time window.
* Users can view a **weekly calendar** per provider to understand availability.
* Rescheduling and cancellation retain a simple status history (e.g., Scheduled, Completed, Canceled, No‑Show).

### 3.4 Visit Documentation

* A clinician can create a **visit note** only after an appointment has occurred.
* Each appointment can produce at most one visit note.
* A visit note includes key clinical details (high‑level, team to define).
* Visit notes are viewable and editable until the related bill is finalized.

### 3.5 Billing & Payments

* Staff can **generate a bill** from an existing visit. The bill includes one or more line items and a computed total.
* Generating a bill for the same visit more than once will not create duplicates (returns the already generated bill).
* Staff can record **payments** against a bill using at least two payment methods (e.g., cash, card as simulated).
* The system prevents over‑payment and shows remaining balance.
* Users can list **outstanding balances** and see **daily collections**.

### 3.6 Reporting (Basic)

* Provide an “Outstanding balances” view/report with filters (e.g., by person/provider).
* Provide a “Daily collections” view/report filtered by date.
* Users can export report results to CSV.

### 3.7 Notifications (Later Phase)

* When a new appointment is created, send a notification to the relevant party/parties.
* When a bill is finalized, send a notification to the relevant party/parties.
* Support at least two notification channels (e.g., email and SMS simulated) with simple **preferences** per recipient.

### 3.8 Audit & Activity

* Every create/update/delete is recorded with who performed it, when, and a brief description of what changed.
* Key system decisions (e.g., double‑booking prevented) are traceable via logs or activity views.

## 4) Usability & Behavior (Functional Expectations)

* Forms provide clear validation messages and prevent submission until required fields are valid.
* Lists are paginated and searchable; search is case‑insensitive.
* Time‑based features handle the clinic’s local time correctly and communicate it in the UI.
* Errors are shown in plain language with actionable next steps (e.g., “conflicts with an existing appointment at …”).

## 5) Reporting & Exports (Functional)

* Reports show totals and subtotals where applicable.
* Exports reflect precisely what is shown in the UI (respect filters and ordering).

## 6) Notifications (Functional)

* Users can opt in/out per channel (email/SMS simulated) on their profile or settings page. (You have to simulate/mock this functionality)
* If a notification fails, the system retries later and avoids sending duplicates for the same event.

## 7) Acceptance of the Whole Project (End‑to‑End Flows)

To consider the capstone functionally complete, the following flows must work from the UI:  
1. **Registration Flow:** create a person record → find it via search → view details.  
2. **Scheduling Flow:** open provider calendar → create appointment → attempt conflicting booking (should be blocked) → reschedule/cancel.  
3. **Clinical Flow:** mark appointment done → create visit note → view/edit visit note.  
4. **Billing Flow:** generate bill from visit → record partial payment → list outstanding balances → record remaining payment.  
5. **Reporting Flow:** view daily collections for a chosen date → export CSV.  
6. **Notification Flow:** create appointment and finalize a bill → notifications are recorded as sent (simulated channels) with no duplicates.

## 9) Phase Reference

Detailed **phase scope and user stories** are provided in a separate document: *CareLite — Phase User Stories (Functional Guide).*