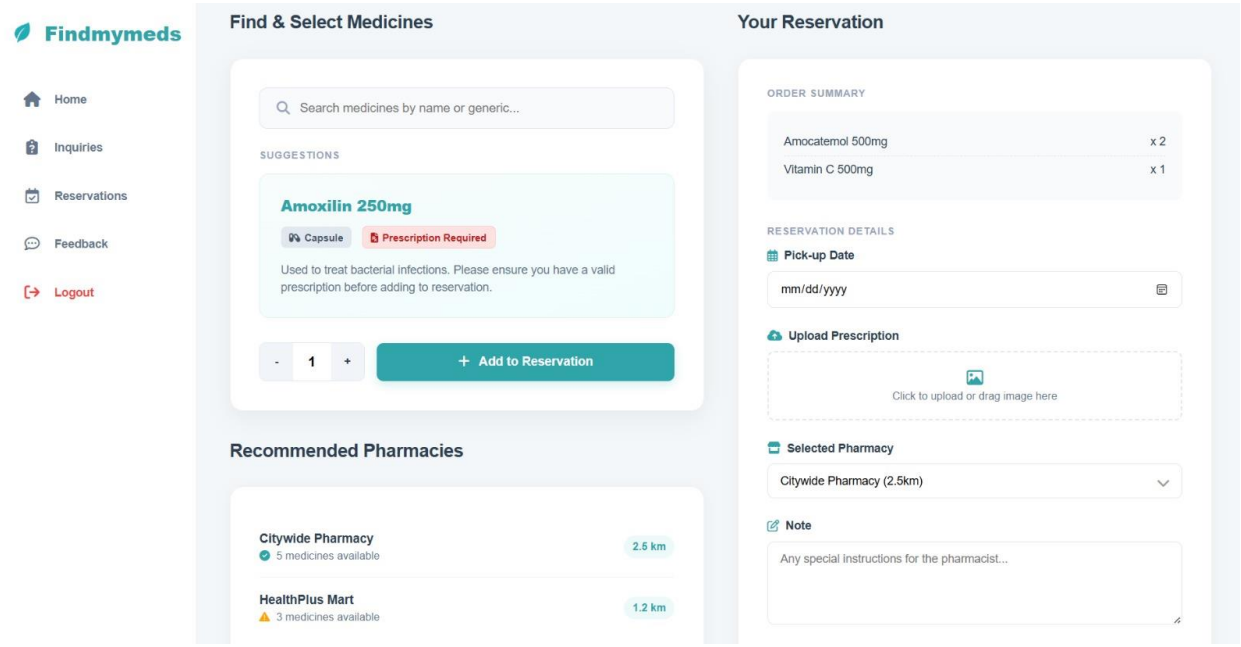
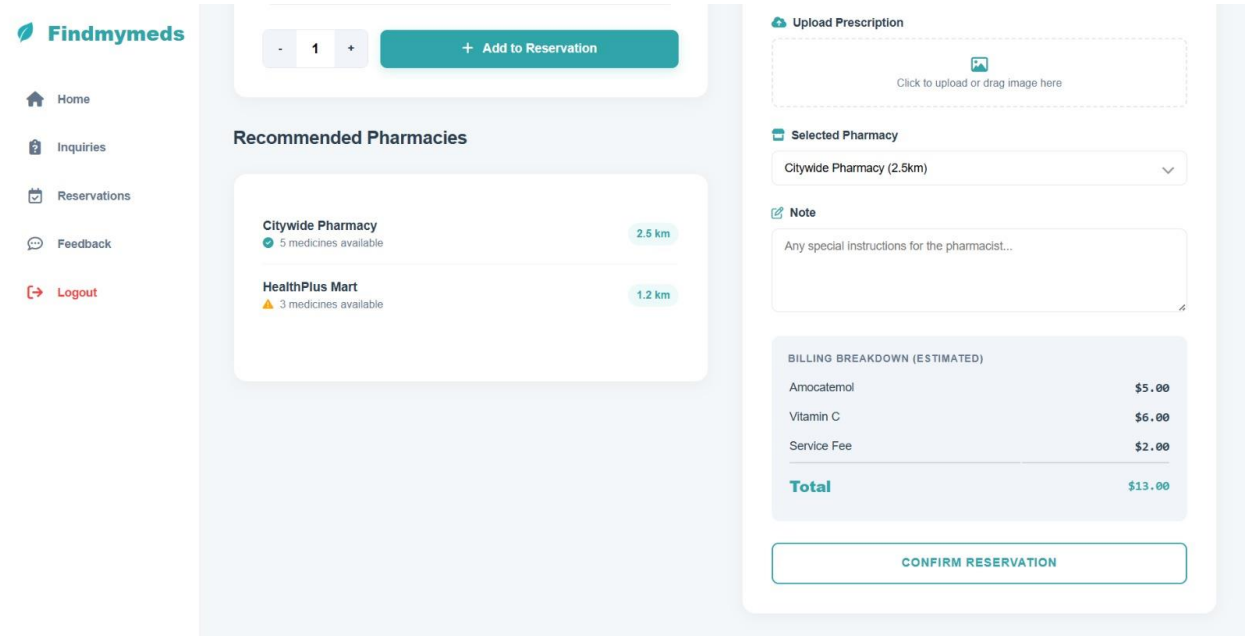
Reservation Page

Civilian-reservation.html – Patient Tech / Reservation Page (Revised & Final)

You have html code of this and follow it…





📌 Core Purpose

This page enables civilians to **ethically reserve medicines** by:

- Selecting **what they need**
- Declaring **how much they need**
- Allowing the system to **recommend fair pharmacies**
- Finalizing a **traceable, accountable reservation**

This is **not checkout chaos** — it's controlled healthcare logic.

🧭 Access Points

- Drug Dictionary → *Reserve Medicine*
- Dashboard → *Make Reservation*
- Recommended Pharmacy → *Reserve*

Only logged-in civilians.

🧩 Page Layout (Single-Flow, Multi-Section)

```
| Left Sidebar | Main Reservation Flow                          |
|              |------------------------------------------------|
|              | 1. Find & Select Medicine                      |
|              |------------------------------------------------|
|              | 2. Recommended Pharmacies                      |
|              |------------------------------------------------|
|              | 3. Your Reservation (Confirmation & Billing)   |
```

🧭 Left Sidebar (Persistent)

Standard civilian Naoba:

- Home
- Dashboard
- Inquiries
- Reservations (active)
- Feedback & Support
- Logout

🧪 SECTION 1: Find & Select Medicine (Updated)

📌 Purpose

Defines **what medicine** and **how much** the user genuinely needs.
This is the **input foundation** for ethical pharmacy recommendation.

📜 Contents

🔍 Medicine Search

- Autocomplete input
- Searches **Medicine Registry**
- Only ACTIVE medicines shown

📋 Selected Medicine Info (Read-Only)

Once selected:

- Medicine Name
- Generic Name
- Category
- Prescription Required (Yes / No)
- Short usage note

🔢 Quantity Selection (Important Change)

This happens HERE, not later.

- Number input:
  - "Required Quantity"
- Minimum: 1
- No maximum at this stage (ethical check happens later)
- Helper text:

  "Select the quantity you require for treatment."

➕ Add to Reservation Button

- Enabled only when:
  - Valid medicine selected
  - Quantity entered

Behavior

1. User selects medicine
2. User selects quantity
3. Clicks **Add to Reservation**
4. System:
   - Locks medicine + quantity
   - Triggers pharmacy recommendation logic

🏪 SECTION 2: Recommended Pharmacies

📌 Purpose

Shows **only pharmacies that can ethically fulfill the selected quantity**.

This section reacts to:

- Selected medicine
- Requested quantity
- User location

🧠 Filtering Logic (Conceptual)

A pharmacy appears **only if**:

- Pharmacy status = ACTIVE
- Medicine available
- Available quantity ≥ requested quantity

This prevents:

- Partial fulfillment
- False reservations
- Stock abuse

📜 Pharmacy Card Contents

Each card displays:

- Pharmacy Name
- Location
- Available Quantity
- Distance (optional)
- Availability badge (e.g. “Fully Available”)
- **Select Pharmacy** button

⚙️ Behavior

- Pharmacies load automatically
- Sorted by:
  - Proximity
  - Higher stock (ethical preference)
- User selects **one pharmacy**
- Selection locks pharmacy
- Reservation section becomes active

📄 SECTION 3: Your Reservation (Finalization)

📌 Purpose

This section confirms **intent, responsibility, and compliance**.

📄 Order Summary (Visible Always)

- Medicine Name
- Requested Quantity
- Selected Pharmacy
- Pharmacy Contact Info

📅 Reservation Details

Pickup Date

- Date picker
- Limited to pharmacy working days
- Auto-calculated expiry window shown

📄 Upload Prescription (Conditional)

- Required **only if** medicine requires prescription
- Accepts:
  - Image / PDF
- Mandatory validation before confirm

📝 Additional Notes

- Optional message to pharmacy
- Special instructions
- Character-limited

💳 Billing Summary (Visible but Transparent)

This section is **informational**, not payment-gated.

- Unit price (if applicable)
- Quantity
- Estimated total
- Note:

  > “Final payment is completed at the pharmacy.”

No online payment enforced unless future upgrade.

☑️ Confirmation Controls

- Confirmation checkbox:
  - “I confirm this reservation and will collect before expiry”
- **Confirm Reservation** button
- **Cancel** button

⚙️ Final Behavior on Confirm Reservation

Backend Process Triggered

1. Reservation record created

2. Status set to:
3. `PENDING`
4. Reservation sent to:
    - Selected pharmacy dashboard
5. Pharmacy inventory:
    - Soft-reserved (not deducted yet)
6. User redirected to:
    - **Your Activity Page**

🔁 What Happens After Reservation (Very Important)

At Pharmacy Side

- Pharmacy sees reservation as **Pending**
- Pharmacy can:
    - Approve
    - Reject (with reason)

If Approved

- Status → `APPROVED`
- User notified
- Pickup allowed within expiry window

If Rejected

- Status → `REJECTED`
- Reason visible to user
- Inventory unlocked

If User Collects

- Status → `COLLECTED`

- Inventory deducted
- Reservation moved to history

If User Fails to Collect

- Status → `EXPIRED`
- Inventory unlocked automatically

🧠 Final Mental Model (This Page)

- Quantity declared **before** pharmacy selection
- System ensures **full availability**
- Pharmacy gets clear, ethical requests
- User stays accountable
- No fake reservations
- No partial stock chaos

Backend Implementation – Reservation (Patient Tech)

Explained from ZERO → WORKING SYSTEM

🧠 FIRST: Understand the Big Picture (Very Important)

Your reservation backend does **ONE MAIN JOB**:

Take a user's medicine + quantity
→ find ethical pharmacies
→ lock inventory
→ send a pending reservation to pharmacy
→ track what happens next

Everything else is just **small steps around this**.

🧩 Core Database Tables (Foundation)

You **must** have these tables. No shortcuts.

1️⃣ Medicine Registry (Already exists)

This tells **what medicines are valid**.

```
medicine_registry
- medicine_id (PK)
- medicine_name
- prescription_required
- status (ACTIVE)
```

2️⃣ Pharmacy Table

```
pharmacy
- pharmacy_id (PK)
- pharmacy_name
- location
- status (ACTIVE)
```

3️⃣ Pharmacy Inventory (Very Important)

This tells **what pharmacy has what and how much**.

```
pharmacy_inventory
- inventory_id (PK)
- pharmacy_id (FK)
- medicine_id (FK)
- quantity
```

4️⃣ Reservation Table (MAIN LOGIC TABLE)

This table stores **everything about reservations**.

```
reservation
- reservation_id (PK)
- civilian_id (FK)
- pharmacy_id (FK)
- medicine_id (FK)
- quantity
- pickup_date
- prescription_file (nullable)
```

```
- notes (nullable)
- status
- created_at
- expiry_date
```

Statuses:

```
PENDING
APPROVED
REJECTED
COLLECTED
EXPIRED
```

🧭 BACKEND FLOW OVERVIEW (One Line)

```
Search medicine
→ select quantity
→ find pharmacies
→ select pharmacy
→ confirm reservation
→ send to pharmacy as PENDING
```

Now we break this into **actual backend steps**.

STEP-BY-STEP IMPLEMENTATION

STEP 1: Medicine Search API

Why?

Frontend needs to know **which medicines exist**.

Endpoint

```
GET /api/medicines/search?name=para
```

Backend Logic

1. Receive medicine name
2. Search `medicine_registry`
3. Return matching medicines

Simple Logic

```
SELECT medicine_id, medicine_name, prescription_required
FROM medicine_registry
WHERE medicine_name LIKE '%para%'
AND status = 'ACTIVE';
```

Frontend uses this to show suggestions.

STEP 2: Quantity Selection (No Backend Yet)

At this stage:

- Backend does nothing
- Quantity stays in frontend state

This is **important**:
❌ Don't validate stock here
✅ Stock check happens later

STEP 3: Find Recommended Pharmacies (Critical Step)

Endpoint

```
POST /api/reservations/recommend-pharmacies
```

Input

```
{
  "medicineId": 12,
  "requiredQuantity": 10,
  "userLocation": "Colombo"
}
```

Backend Logic (Slow + Clear)

1. Get medicineId + quantity
2. Search **pharmacy_inventory**
3. Filter pharmacies where:
    - quantity >= requiredQuantity
    - pharmacy.status = ACTIVE

4. Sort pharmacies:
    - nearest first
    - higher stock preferred

SQL Logic

```
SELECT p.pharmacy_id, p.pharmacy_name, pi.quantity
FROM pharmacy_inventory pi
JOIN pharmacy p ON p.pharmacy_id = pi.pharmacy_id
WHERE pi.medicine_id = :medicineId
AND pi.quantity >= :requiredQuantity
AND p.status = 'ACTIVE';
```

Output

```
[
  {
    "pharmacyId": 3,
    "pharmacyName": "City Pharmacy",
    "availableQuantity": 50
  }
]
```

Frontend shows these as **Recommended Pharmacies**.

STEP 4: User Selects Pharmacy (Still No DB Change)

At this moment:

- Backend does nothing
- Frontend just stores selected pharmacyId

This prevents **unnecessary database locking**.

STEP 5: Confirm Reservation (MOST IMPORTANT STEP)

Endpoint

```
POST /api/reservations/confirm
```

Input Payload

```
{
  "medicineId": 12,
  "pharmacyId": 3,
  "quantity": 10,
  "pickupDate": "2025-01-05",
  "notes": "Urgent",
  "prescriptionFile": "file.pdf"
}
```

Backend Logic (This Is Where Everything Happens)

1 Authenticate User

- Get `civilian_id` from token
- If not logged in → reject

---

2 Re-check Inventory (VERY IMPORTANT)

Never trust frontend.

```
SELECT quantity
FROM pharmacy_inventory
WHERE pharmacy_id = :pharmacyId
AND medicine_id = :medicineId;
```

If quantity < requested → reject reservation

---

3 Create Reservation Record

```
INSERT INTO reservation (
  civilian_id,
  pharmacy_id,
  medicine_id,
  quantity,
  pickup_date,
  prescription_file,
  notes,
  status,
  created_at,
  expiry_date
)
VALUES (
  :civilianId,
  :pharmacyId,
  :medicineId,
  :quantity,
  :pickupDate,
  :file,
```

```
  :notes,
  'PENDING',
  NOW(),
  pickupDate + INTERVAL '2 DAYS'
);
```

---

4 Soft Lock Inventory (Optional but Recommended)

```
UPDATE pharmacy_inventory
SET quantity = quantity - :quantity
WHERE pharmacy_id = :pharmacyId
AND medicine_id = :medicineId;
```

(This prevents double booking)

---

5 Notify Pharmacy

- Add notification record
- Pharmacy dashboard shows **PENDING reservation**

Response

```
{
  "message": "Reservation placed successfully",
  "status": "PENDING"
}
```

STEP 6: After Reservation – What Happens Automatically

Pharmacy Side

- Pharmacy reviews reservation
- Approves or rejects

If Pharmacy Approves

```
Status → APPROVED
```

User can collect medicine.

If Pharmacy Rejects

```
Status → REJECTED
Inventory restored
```

STEP 7: Pickup or Expiry (Background Job)

Scheduled Job (Daily)

```
UPDATE reservation
SET status = 'EXPIRED'
WHERE status = 'APPROVED'
AND expiry_date < NOW();
```

Restore inventory.

STEP 8: Activity Page Uses This Data

Your **Your Activity page** simply reads from:

```
reservation table
```

Grouped by:

- PENDING / APPROVED → Active
- COLLECTED / EXPIRED → History

🧠 FINAL SIMPLE MENTAL MODEL (Memorize This)

- Medicine table = what exists
- Inventory table = who has it
- Reservation table = who asked for it
- Status = truth of the system

🏁 Final Words (Important)

You’re not expected to **code this instantly**.
You are expected to **understand the flow**.

Once this logic is clear:

- Coding becomes mechanical
- Bugs reduce
- Ethics stay intact