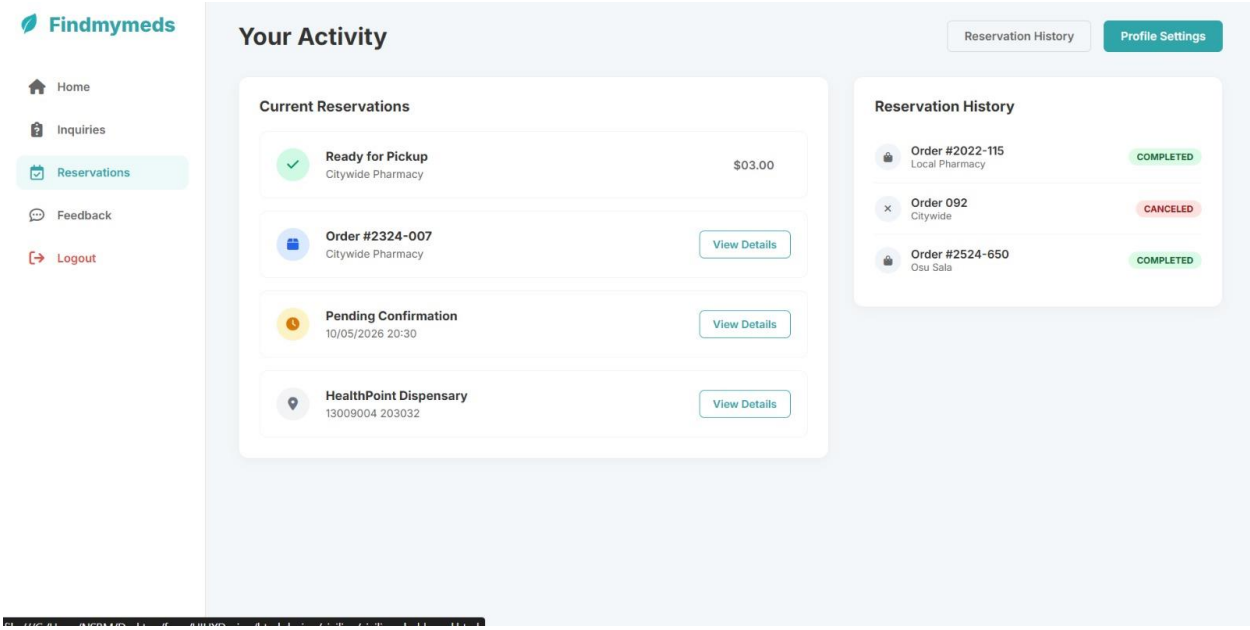
civilian-activity.html – Your Activity Page

You have html code of this too…



📌 Purpose

The **Your Activity** page gives a civilian a **complete, chronological view of everything they've done in the system**.

Think of it as:

- Your personal audit trail
- Your medicine interaction history
- Your "what's happening right now vs what already happened" screen

This page is **read + manage**, not just read.

🧭 How This Page Is Accessed

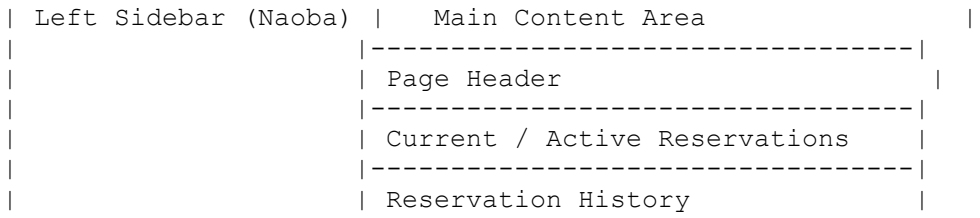
Entry Points

- From **Left Sidebar (Naoba)**
  - "Your Activity" or

    - “Reservations” → “View Activity” (optional shortcut)
- From **Dashboard Card**
    - “Reservations” / “Activity Summary”
- After completing a reservation (redirect option)

Only **logged-in civilians** can access this page.

🧩 Page Layout (High-Level)



🧭 Left Sidebar / Naoba (Persistent)

Same common sidebar used across civilian pages.

Items

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

📜 Page Header Section

Content

- Title: **Your Activity**
- Subtitle: *Track your reservations and history*

Optional Info

- Total reservations count

- Active reservations count

🧩 Main Content Sections

The page is split into **two primary sections**.

🟢 1️⃣ Current / Active Reservations Section

Purpose

Shows **ongoing or actionable reservations**.

These are reservations that:

- Are pending approval
- Are approved but not yet collected
- Are awaiting user action

Content per Reservation Card

Each reservation card displays:

- Reservation ID
- Medicine Name
- Pharmacy Name
- Reserved Quantity
- Reservation Date
- Reservation Status:
  - Pending
  - Approved
  - Rejected
  - Ready for Pickup
- Expiry Date / Pickup Deadline

Actions (Context-Based)

Depending on status:

- **View Details**
- **Cancel Reservation** (if still pending)
- **Contact Pharmacy**
- **Mark as Collected** (optional flow)

Behavior

- Real-time or near-real-time updates
- Automatically moves to history once completed or expired
- Highlight urgent reservations (near expiry)

🔵 2️⃣ Reservation History Section

Purpose

Provides a **read-only log** of all past reservations.

This includes:

- Completed
- Cancelled
- Rejected
- Expired reservations

Content per History Entry

- Reservation ID
- Medicine Name
- Pharmacy Name
- Quantity
- Reservation Date
- Completion Date
- Final Status

Filters & Sorting (Optional but Recommended)

- Filter by:
  - Date range
  - Status
  - Pharmacy
- Sort by:
  - Latest first
  - Oldest first

Behavior

- Read-only
- No destructive actions
- Pagination for large histories

🔍 Empty States Handling

No Current Reservations

- Show message:

  > “You don’t have any active reservations.”

No Reservation History

- Show message:

  > “No past reservations found.”

🧩 Components Summary

UI Components

- Persistent Sidebar
- Page Header
- Reservation Cards
- Status Badges
- Action Buttons
- Filters & Pagination

Backend Dependencies

- Reservation Table
- Pharmacy Table
- Medicine Registry Table

🔁 User Flow (Simple)

1. Civilian opens **Your Activity**
2. Sees:
    - Active reservations (top)
    - History (bottom)
3. Takes actions on active reservations
4. Completed items move automatically to history

🧠 Final Mental Model (For Teammates)

- **Top = What still matters**
- **Bottom = What already happened**
- No duplicate data
- No confusion
- One page = full reservation story

Backend Implementation – civilian-activity.html (Your Activity Page)

This backend is about **reading, categorizing, and safely managing reservation data**.

1️⃣ Core Tables Involved

🔷 Reservation Table (MAIN)

reservation

- reservation_id (PK)
- civilian_id (FK)
- pharmacy_id (FK)
- medicine_id (FK)
- quantity
- reservation_status
  (`PENDING`, `APPROVED`, `REJECTED`, `READY`, `COLLECTED`, `CANCELLED`, `EXPIRED`)
- reservation_date
- expiry_date
- completed_date (nullable)

🔷 Medicine Registry Table

medicine_registry

- medicine_id
- medicine_name

🔷 Pharmacy Table

pharmacy

- pharmacy_id
- pharmacy_name
- contact_number

---

2 Access Control (Mandatory)

- Endpoint accessible **only to authenticated civilians**
- `civilian_id` is taken from **JWT / session**
- Civilian can only see **their own reservations**

No query parameter should accept civilian_id directly.

---

3 API Design Overview

Main Endpoint

```
GET /api/civilian/activity
```

This single endpoint returns:

- Active reservations
- Reservation history

---

4 Step-by-Step Backend Logic

STEP 1: Identify Logged-In Civilian

Logic

- Extract `civilian_id` from token/session
- If missing → `401 Unauthorized`

STEP 2: Fetch Current / Active Reservations

Definition of "Active"

Statuses included:

- `PENDING`
- `APPROVED`
- `READY`

SQL-like Logic

```
SELECT
  r.reservation_id,
  m.medicine_name,
  p.pharmacy_name,
  r.quantity,
  r.reservation_date,
  r.expiry_date,
```

```
  r.reservation_status
FROM reservation r
JOIN medicine_registry m ON m.medicine_id = r.medicine_id
JOIN pharmacy p ON p.pharmacy_id = r.pharmacy_id
WHERE r.civilian_id = :civilian_id
AND r.reservation_status IN ('PENDING', 'APPROVED', 'READY')
ORDER BY r.reservation_date DESC;
```

STEP 3: Fetch Reservation History

Definition of “History”

Statuses included:

- COLLECTED
- REJECTED
- CANCELLED
- EXPIRED

SQL-like Logic

```
SELECT
  r.reservation_id,
  m.medicine_name,
  p.pharmacy_name,
  r.quantity,
  r.reservation_date,
  r.completed_date,
  r.reservation_status
FROM reservation r
JOIN medicine_registry m ON m.medicine_id = r.medicine_id
JOIN pharmacy p ON p.pharmacy_id = r.pharmacy_id
WHERE r.civilian_id = :civilian_id
AND r.reservation_status IN ('COLLECTED','REJECTED','CANCELLED','EXPIRED')
ORDER BY r.completed_date DESC;
```

STEP 4: Build Response DTO

Unified Response

```
{
  "activeReservations": [
    {
      "reservationId": 1021,
      "medicineName": "Paracetamol",
      "pharmacyName": "HealthPlus Pharmacy",
      "quantity": 10,
      "reservationDate": "2025-12-20",
      "expiryDate": "2025-12-23",
```

```
      "status": "READY"
    }
  ],
  "reservationHistory": [
    {
      "reservationId": 987,
      "medicineName": "Amoxicillin",
      "pharmacyName": "City Care Pharmacy",
      "quantity": 5,
      "reservationDate": "2025-11-12",
      "completedDate": "2025-11-14",
      "status": "COLLECTED"
    }
  ]
}
```

Frontend renders **two sections from one response**.

---

5 Action-Based Endpoints (Optional but Needed)

🔷 Cancel Reservation

`PUT /api/civilian/reservation/{id}/cancel`

Rules

- Only `PENDING` allowed
- Ownership check mandatory

Status update:

`PENDING → CANCELLED`

🔷 Mark as Collected

`PUT /api/civilian/reservation/{id}/collect`

Rules

- Only `READY`
- Sets:
  - status → `COLLECTED`
  - completed_date → NOW()

---

6 Auto-State Transitions (Background Job)

Expiry Handling (Scheduled Task)

Runs periodically:

```
IF current_date > expiry_date
AND status IN (PENDING, APPROVED, READY)
→ status = EXPIRED
```

This keeps **Activity page clean & accurate**.

---

7 Indexing & Performance

Required Indexes

- reservation.civilian_id
- reservation.reservation_status
- reservation.reservation_date

---

8 Edge Case Handling

No Active Reservations

Return empty array → frontend shows friendly message

Large History

- Pagination supported:

```
GET /api/civilian/activity?historyPage=1
```

---

9 Clean Mental Model (For Dev Team)

- **One civilian → many reservations**
- Status defines where it appears:
  - Active section
  - History section
- Backend decides categories

- Frontend only renders

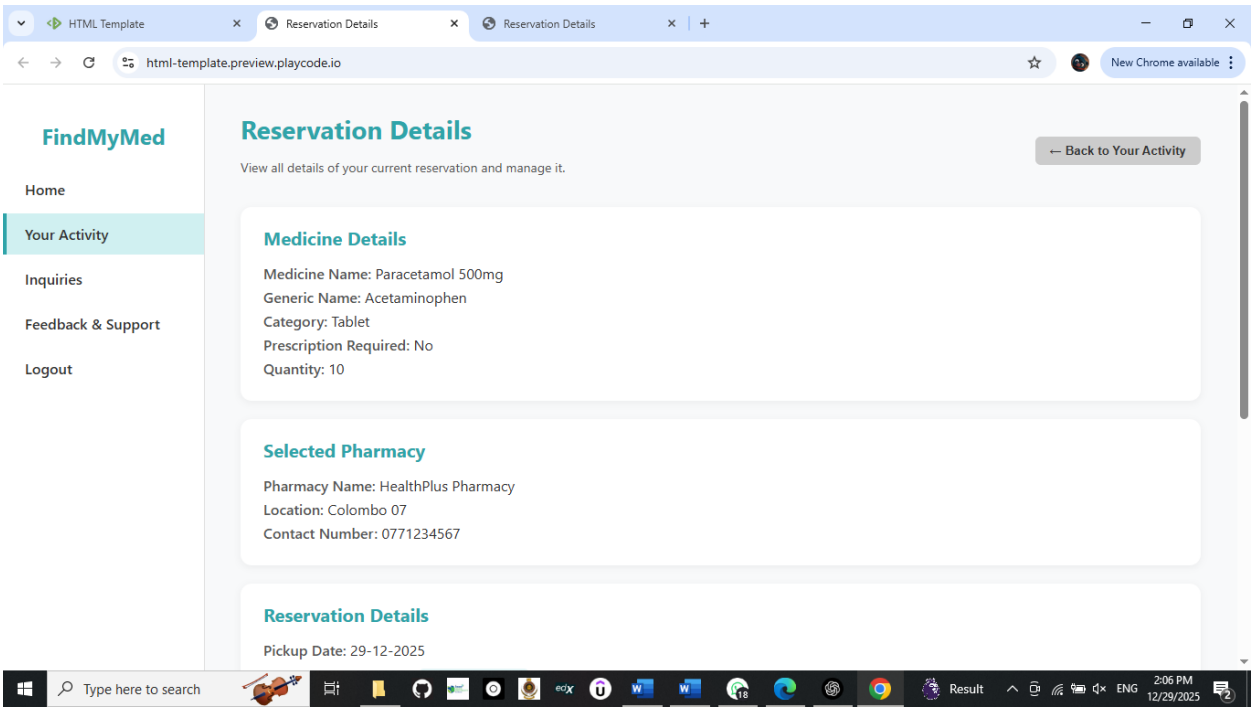
🔚 Final Summary

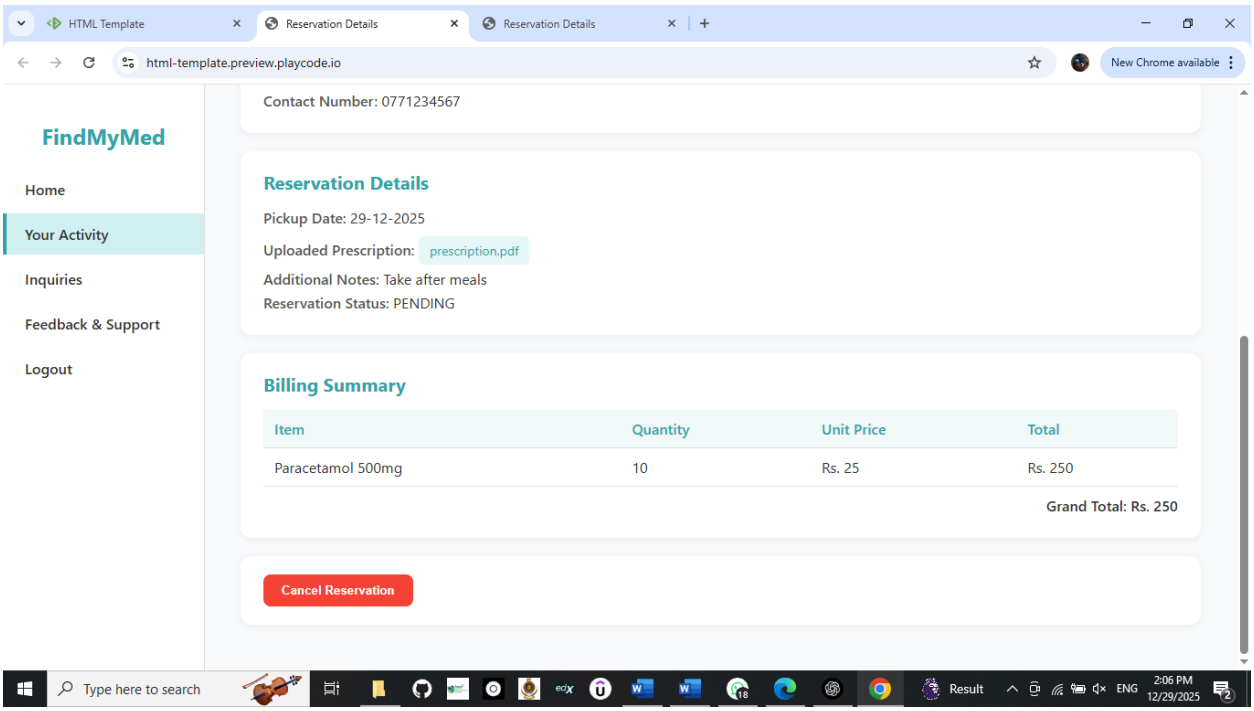
- One endpoint powers the whole page
- Status-based filtering is the key
- Strict ownership checks
- Background jobs maintain truth

View Details of Current Reservation Page – Full Explanation

📌 Purpose

This page allows a civilian user to **view all details of a specific reservation** they have made. It's a **read-only page** with some management actions (like canceling a reservation).
It ensures transparency and lets the user track reservation status, pickup details, and billing information.





🧭 How This Page Can Be Accessed

- From the **Activity Page** (`civilian-activity.html`)
- Locate the **Current Reservations Table**
- Click the **"View Details" button** next to any reservation
- The system redirects to this **Reservation Details Page**, pre-filled with all relevant data for that specific reservation

Only logged-in civilian users can access their own reservation details.

🧩 Page Layout & Sections

The page follows the **same design style as other civilian pages** (sidebar navigation, color palette, rounded cards, modern readable font).

The page is **split into the following sections**:

---

1 Page Header

- **Title:** "Reservation Details"
- **Subtitle / description:** "View all details of your current reservation and manage it"
- **Action Button:**
    - **Back to Your Activity** → navigates back to the Activity page

---

2 Medicine Details Section

- **Purpose:** Show information about the medicine being reserved
- **Content bullet-wise:**
    - Medicine Name
    - Generic Name
    - Category (Tablet, Syrup, etc.)
    - Prescription Required (Yes / No)
    - Quantity reserved

---

3 Selected Pharmacy Section

- **Purpose:** Display the pharmacy assigned for this reservation
- **Content bullet-wise:**
    - Pharmacy Name
    - Location (City / Area)
    - Contact Number

---

4 Reservation Details Section

- **Purpose:** Show reservation-specific information
- **Content bullet-wise:**
    - Pickup Date
    - Uploaded Prescription (downloadable link or preview if applicable)
    - Additional Notes (optional text added by user)
    - Reservation Status (Pending / Approved / Collected / Expired / Cancelled)

---

5 Billing Summary Section

- **Purpose:** Provide an overview of billing or price estimation

- **Content bullet-wise:**
  - Item Name (Medicine Name)
  - Quantity
  - Unit Price
  - Total
  - Grand Total

6️⃣ Actions Section

- **Purpose:** Allow the user to manage the reservation
- **Action buttons:**
  - **Cancel Reservation:**
    - Allows user to cancel the reservation at any time before collection
    - Updates reservation status in the backend (`CANCELLED`)
    - Releases reserved stock in pharmacy inventory
  - **Back to Activity:** (already in header, but can also be repeated at the bottom)
    - Returns to Activity page

No other edits are allowed — this is a **view/manage-only** page, not a reservation creation or edit page.

⚙️ Page Behavior Summary

- Data is **pulled from the database** based on the reservation ID
- **Reservation Status:** updates dynamically if any change occurs (e.g., pharmacy approves or user cancels)
- **Cancel Reservation button:** only enabled if status allows cancellation (Pending / Approved but not yet collected)
- **Back button:** ensures easy navigation to the Activity page

---

🧠 Summary / Flow

1. User goes to Activity page → Current Reservations table
2. Clicks **View Details**
3. Reservation Details Page loads with:
   - Medicine info
   - Pharmacy info
   - Reservation info (pickup, prescription, notes, status)
   - Billing summary

4. User can:
    - **Cancel reservation** (if allowed)
    - **Go back** to Activity page
5. The backend ensures **inventory consistency**, **reservation status updates**, and **ethical reservation tracking**

Backend Implementation – Reservation Details Page

🧠 Purpose

- Fetch all details of a **specific reservation** for the logged-in civilian
- Ensure **data integrity** (user can only view their own reservations)
- Provide the ability to **cancel the reservation** (if allowed)
- Return all related info: medicine, pharmacy, pickup, prescription, notes, billing, and status

---

STEP 1️⃣ API Endpoint Design

Endpoint:

```
GET /api/reservations/:reservationId
```

Purpose: Fetch reservation details by reservation ID.

Parameters:

- `reservationId` → The unique ID of the reservation (from the "View Details" button)

Authentication:

- Must validate the logged-in civilian user (token/session)
- Ensure the reservation belongs to the logged-in user

STEP 2️⃣ Validate Reservation Access

1. Receive `reservationId` and user session info (`civilianId`)
2. Query **reservation table** to check:

```
SELECT * FROM reservation
WHERE reservation_id = :reservationId
AND civilian_id = :civilianId;
```

3. If no result → return **403 Forbidden** or error:

"You are not allowed to view this reservation."

STEP 3 – Fetch Related Data

a) Medicine Details

```
SELECT medicine_name, generic_name, category, prescription_required
FROM medicine_registry
WHERE medicine_id = :medicineId;
```

b) Pharmacy Details

```
SELECT pharmacy_name, location, contact_number
FROM pharmacy
WHERE pharmacy_id = :pharmacyId;
```

c) Reservation Details

- From `reservation` table:
  - quantity
  - pickup_date
  - prescription_file
  - notes
  - status

d) Billing Info (if applicable)

- Unit price, quantity, total calculation
- Can be stored in reservation table or dynamically calculated from inventory/pricing table

---

STEP 4 – Prepare Response

Backend Response Structure:

```
{
  "reservationId": 101,
  "medicine": {
```

```
    "name": "Paracetamol 500mg",
    "genericName": "Acetaminophen",
    "category": "Tablet",
    "prescriptionRequired": false,
    "quantity": 10
  },
  "pharmacy": {
    "name": "HealthPlus Pharmacy",
    "location": "Colombo 07",
    "contact": "0771234567"
  },
  "reservationDetails": {
    "pickupDate": "2025-12-29",
    "prescriptionFile": "prescription.pdf",
    "notes": "Take after meals",
    "status": "PENDING"
  },
  "billing": {
    "unitPrice": 25,
    "total": 250,
    "grandTotal": 250
  }
}
```

Frontend can now **render the reservation details page** using this JSON.

STEP 5 – Cancel Reservation API

Endpoint:

```
POST /api/reservations/:reservationId/cancel
```

Backend Logic:

1. Validate user access (civilianId matches reservation)
2. Check if status allows cancellation (Pending or Approved but not collected)
3. Update reservation status:

```
UPDATE reservation
SET status = 'CANCELLED'
WHERE reservation_id = :reservationId
AND civilian_id = :civilianId;
```

4. Restore inventory:

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

5. Return success message to frontend

```
{
  "message": "Reservation cancelled successfully",
  "status": "CANCELLED"
}
```

STEP 6 — Optional: Real-Time Status Updates

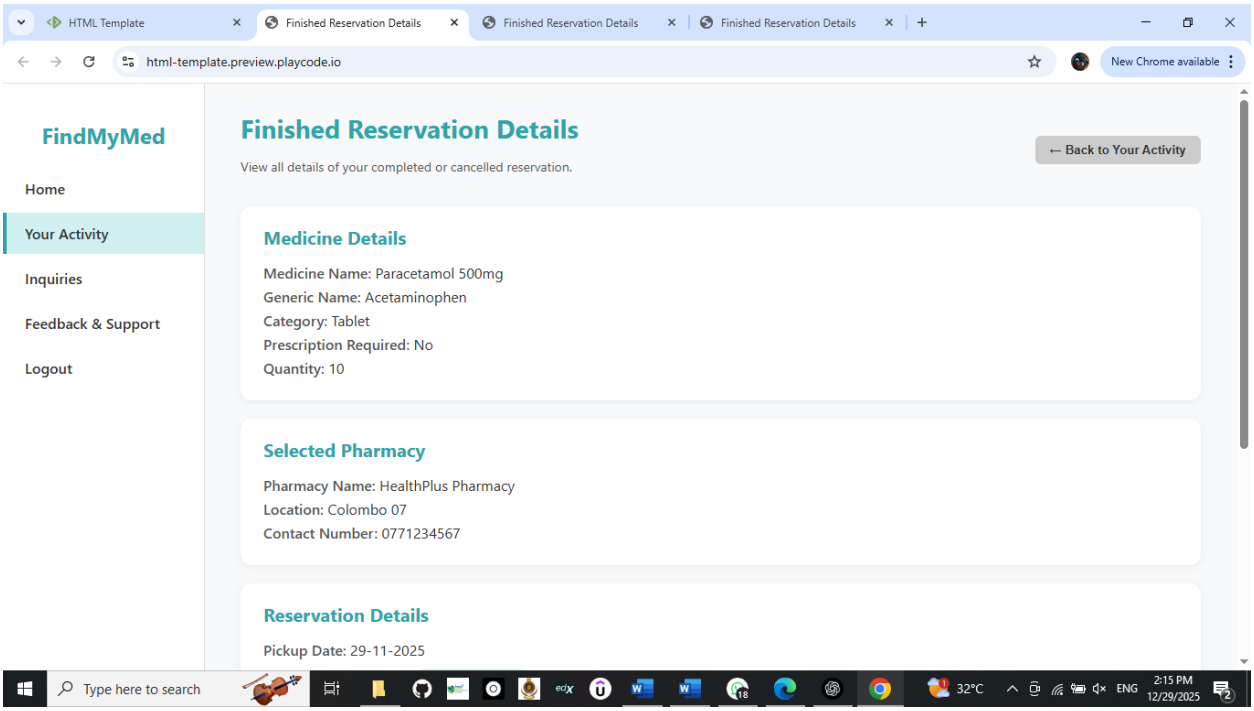
- If you want **live updates** (e.g., status changes by pharmacy):
  - Use WebSockets or polling
  - Update reservation status on page without reload

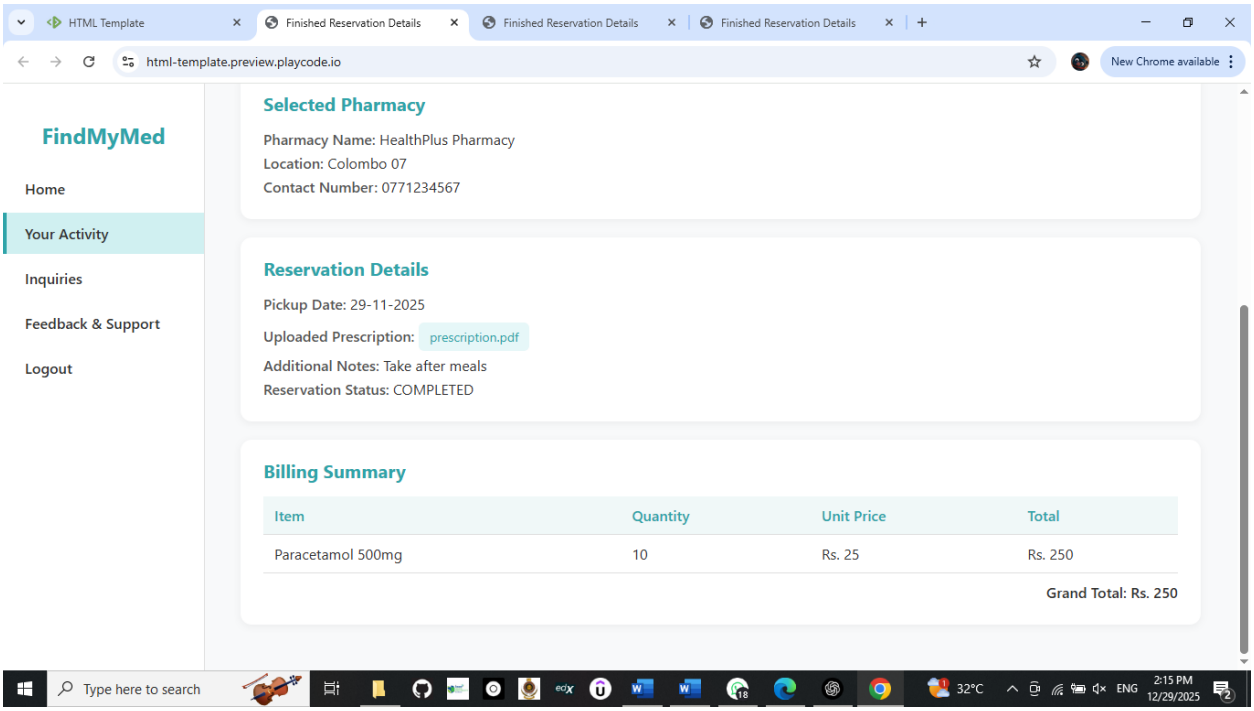
STEP 7 — Summary / Flow

1. User clicks **View Details** → frontend calls `GET /api/reservations/:reservationId`
2. Backend validates user → fetches medicine, pharmacy, reservation, billing info → returns JSON
3. Frontend renders page (all sections: medicine, pharmacy, reservation, billing)
4. **Cancel button:** frontend calls `POST /api/reservations/:reservationId/cancel` → backend updates status & inventory → return updated status

This fully completes the backend logic for the **Reservation Details Page**, including viewing and cancellation.

Finished Reservation Details Page – Full Explanation


HTML Template
Finished Reservation Details
Finished Reservation Details
Finished Reservation Details
html-template.preview.playcode.io
New Chrome available
FindMyMed
Home
Your Activity
Inquiries
Feedback & Support
Logout
Finished Reservation Details
View all details of your completed or cancelled reservation.
← Back to Your Activity
Medicine Details
Medicine Name: Paracetamol 500mg
Generic Name: Acetaminophen
Category: Tablet
Prescription Required: No
Quantity: 10
Selected Pharmacy
Pharmacy Name: HealthPlus Pharmacy
Location: Colombo 07
Contact Number: 0771234567
Reservation Details
Pickup Date: 29-11-2025
Type here to search
32°C
ENG
2:15 PM
12/29/2025


HTML Template
Finished Reservation Details
Finished Reservation Details
Finished Reservation Details
html-template.preview.playcode.io
New Chrome available
FindMyMed
Home
Your Activity
Inquiries
Feedback & Support
Logout
Selected Pharmacy
Pharmacy Name: HealthPlus Pharmacy
Location: Colombo 07
Contact Number: 0771234567
Reservation Details
Pickup Date: 29-11-2025
Uploaded Prescription: prescription.pdf
Additional Notes: Take after meals
Reservation Status: COMPLETED
Billing Summary
Item
Quantity
Unit Price
Total
Paracetamol 500mg
10
Rs. 25
Rs. 250
Grand Total: Rs. 250
Type here to search
32°C
ENG
2:15 PM
12/29/2025

📌 Purpose

This page allows a civilian user to **view the details of reservations that are already completed, cancelled, or expired**.

- This page is **read-only**, unlike the Current Reservation Details page.
- Users cannot edit, cancel, or modify any details here.
- It provides a clear historical record of their past activity.

🧭 How This Page Can Be Accessed

- From the **Activity Page** (`civilian-activity.html`)
- Locate the **Finished / Past Reservations Table**
- Click the **"View Details" button** next to any finished or cancelled reservation
- The system redirects to this **Finished Reservation Details Page**

Only logged-in civilian users can access their own finished reservations.

🧩 Page Layout & Sections

The page structure **mirrors the Current Reservation Details Page** for consistency:

1️⃣ Page Header

- **Title:** "Finished Reservation Details"
- **Subtitle:** "View all details of your completed or cancelled reservation"
- **Action Button:**
  - **Back to Your Activity** → navigates back to the Activity page

---

2️⃣ Medicine Details Section

- Shows information about the medicine that was reserved.
  Content bullet-wise:
- Medicine Name
- Generic Name
- Category (Tablet, Syrup, etc.)

- Prescription Required (Yes / No)
- Quantity reserved

---

3 Selected Pharmacy Section

- Shows the pharmacy assigned for this reservation.
  Content bullet-wise:
- Pharmacy Name
- Location (City / Area)
- Contact Number

---

4 Reservation Details Section

- Shows reservation-specific information.
  Content bullet-wise:
- Pickup Date
- Uploaded Prescription (downloadable link or preview if applicable)
- Additional Notes (optional text added by user)
- Reservation Status: COMPLETED / CANCELLED / EXPIRED

Unlike the current reservation page, there are **no action buttons for canceling**.

---

5 Billing Summary Section

- Provides an overview of billing or price estimation.
  Content bullet-wise:
- Item Name (Medicine Name)
- Quantity
- Unit Price
- Total
- Grand Total

---

6 Actions Section

- **Only Back Button is available**
  - Returns the user to the Activity page
- **No Cancel or Edit buttons**, because reservation is finished

---

⚙️ Page Behavior Summary

- Data is **pulled from the database** based on the reservation ID
- Users **cannot modify** any fields
- Provides a **clear historical record** of their past activity
- Reservation Status is displayed but **not editable**
- Backend can automatically **delete records older than 60 days** to keep the database clean

---

🧠 Summary / Flow

1. User goes to Activity page → Finished / Past Reservations Table
2. Clicks **View Details**
3. Finished Reservation Details Page loads with:
    - Medicine info
    - Pharmacy info
    - Reservation info (pickup, prescription, notes, status)
    - Billing summary
4. User can:
    - **Go back** to Activity page
5. No modifications or cancellations are allowed.

✅ This page ensures **historical tracking** and **consistency** while maintaining the system's integrity for completed or cancelled reservations.

Backend Implementation – Finished Reservation Details Page

🧠 Purpose

- Fetch all details of a **finished, expired, or cancelled reservation** for the logged-in civilian.
- Provide a **read-only view** (no cancel/edit actions).
- Support automatic **cleanup of old reservations** (older than 60 days).

STEP 1 – API Endpoint Design

Endpoint:

```
GET /api/reservations/finished/:reservationId
```

Purpose: Fetch finished reservation details by reservation ID.

Parameters:

- `reservationId` → The unique ID of the reservation (from "View Details" button in Finished Reservations Table)

Authentication:

- Validate the logged-in civilian user
- Ensure the reservation belongs to the logged-in user

STEP 2 – Validate Reservation Access

1. Receive `reservationId` and user session info (`civilianId`)
2. Query **reservation table**:

```
SELECT * FROM reservation
WHERE reservation_id = :reservationId
AND civilian_id = :civilianId
AND status IN ('COMPLETED', 'CANCELLED', 'EXPIRED');
```

3. If no result → return **403 Forbidden** or error:

"You are not allowed to view this reservation or it does not exist."

STEP 3 – Fetch Related Data

a) Medicine Details

```
SELECT medicine_name, generic_name, category, prescription_required, quantity
FROM medicine_registry
WHERE medicine_id = :medicineId;
```

b) Pharmacy Details

```
SELECT pharmacy_name, location, contact_number
FROM pharmacy
WHERE pharmacy_id = :pharmacyId;
```

c) Reservation Details

- From `reservation` table:
  - Pickup date
  - Uploaded prescription file
  - Additional notes
  - Status (`COMPLETED`, `CANCELLED`, `EXPIRED`)

d) Billing Info

- Unit price, quantity, total
- Could be pre-calculated and stored in reservation table, or calculated dynamically from inventory/pricing table

---

STEP 4 – Prepare Response

Backend Response Structure:

```
{
  "reservationId": 102,
  "medicine": {
    "name": "Paracetamol 500mg",
    "genericName": "Acetaminophen",
    "category": "Tablet",
    "prescriptionRequired": false,
    "quantity": 10
  },
  "pharmacy": {
    "name": "HealthPlus Pharmacy",
    "location": "Colombo 07",
    "contact": "0771234567"
  },
  "reservationDetails": {
    "pickupDate": "2025-11-29",
    "prescriptionFile": "prescription.pdf",
    "notes": "Take after meals",
    "status": "COMPLETED"
  },
  "billing": {
    "unitPrice": 25,
    "total": 250,
    "grandTotal": 250
  }
}
```

Frontend can render this JSON in the **Finished Reservation Details Page**.

STEP 5 — Automatic Deletion of Old Reservations

Goal: Delete reservation records **older than 60 days** automatically.

a) SQL Query

```
DELETE FROM reservation
WHERE status IN ('COMPLETED', 'CANCELLED', 'EXPIRED')
AND pickup_date < NOW() - INTERVAL 60 DAY;
```

b) Scheduling

- Use a **cron job** or backend scheduler (depending on your backend framework)
- Example: Run **daily** at midnight to clean old records

This keeps the database optimized without affecting current or active reservations.

STEP 6 — Optional: Pagination / History List

- If you display a list of finished reservations on the Activity page:

```
SELECT reservation_id, pickup_date, status
FROM reservation
WHERE civilian_id = :civilianId
AND status IN ('COMPLETED', 'CANCELLED', 'EXPIRED')
ORDER BY pickup_date DESC
LIMIT 50 OFFSET :pageOffset;
```

- Supports **efficient frontend rendering** for long histories

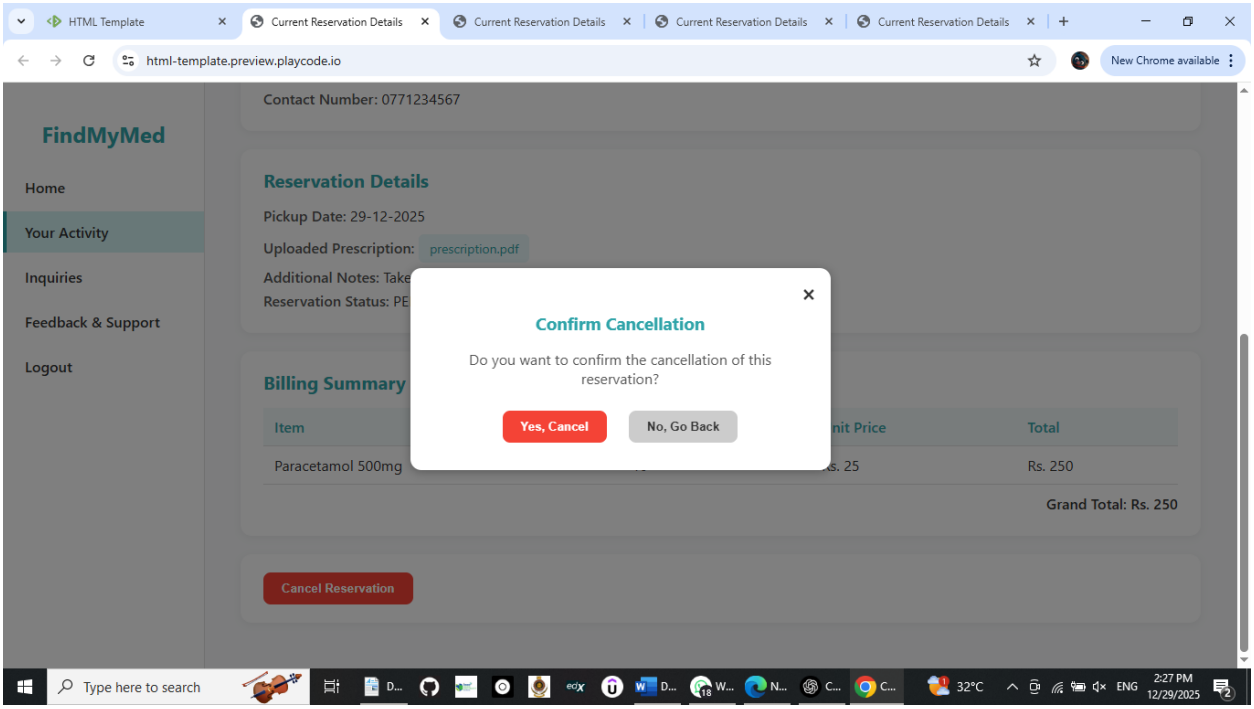
---

STEP 7 — Summary / Flow

1. User clicks **View Details** in Finished Reservations Table
2. Frontend calls `GET /api/reservations/finished/:reservationId`
3. Backend validates user → fetches:
    - Medicine info
    - Pharmacy info
    - Reservation info

    - Billing info
4. Returns JSON → Frontend renders **read-only finished reservation page**
5. **Automatic deletion job** runs daily → removes records older than 60 days

Cancel Reservation Confirmation Modal – Explanation



📌 Purpose

- To **prevent accidental cancellation** of a current reservation.
- It ensures the user **intentionally confirms** the action before the system updates the reservation.
- Provides a **clear, user-friendly confirmation step** consistent with modern UI/UX practices.

🧭 How It Works

1. User clicks the **"Cancel Reservation"** button on the **Current Reservation Details Page**.
2. The modal **pops up**, overlaying the page content with a semi-transparent background.
3. The modal contains:
    - **Header:** "Confirm Cancellation"

- **Message:** “Do you want to confirm the cancellation of this reservation?”
- **Two buttons:**
  - **Yes, Cancel** → triggers the reservation cancellation process
  - **No, Go Back** → closes the modal without changing anything
- **Close (X) button** in the top-right corner for optional dismissal

4. Clicking outside the modal **also closes it**, maintaining a smooth user experience.

🧩 Modal Sections & Content

1️⃣ Header

- Text: **“Confirm Cancellation”**
- Color: Primary system color (#2FA4A9) to match branding
- Purpose: Quickly informs the user about the action

2️⃣ Body / Message

- Text: **“Do you want to confirm the cancellation of this reservation?”**
- Purpose: Clarifies the action and consequences

3️⃣ Action Buttons

- **Yes, Cancel (Red Button)**
  - Color: #f44336 (red for danger / destructive action)
  - Action: Calls backend API to **cancel the reservation**, updates status, optionally restores inventory
- **No, Go Back (Gray Button)**
  - Color: #ccc (neutral)
  - Action: Closes the modal without making any changes

4️⃣ Close Button (X)

- Positioned in the **top-right corner**
- Allows dismissal without clicking the “No” button
- Works the same way as clicking outside the modal

⚙️ Behavior & Flow

1. **Opening the modal:**
   - Triggered via JavaScript when user clicks **Cancel Reservation** button

    - Modal becomes **visible**, page behind is slightly dimmed
2. **Closing the modal:**
    - Clicking **No, Go Back**
    - Clicking the **X**
    - Clicking outside the modal
3. **Confirming cancellation:**
    - Clicking **Yes, Cancel**
    - JavaScript closes the modal and triggers backend cancellation logic
    - Optionally shows a **success message** (e.g., "Reservation cancelled successfully!")
    - Frontend can then **redirect** back to the Activity page or update the page dynamically

---

🔑 Key Points for Your Team

- The modal is **purely a UI/UX safety feature** to confirm destructive actions.
- Styling and placement match the system's **color palette and card-based design**.
- Can easily be **connected to backend APIs** to make cancellation permanent.
- Reusable pattern: Can be adapted for other destructive actions (e.g., deleting feedback, reservations).