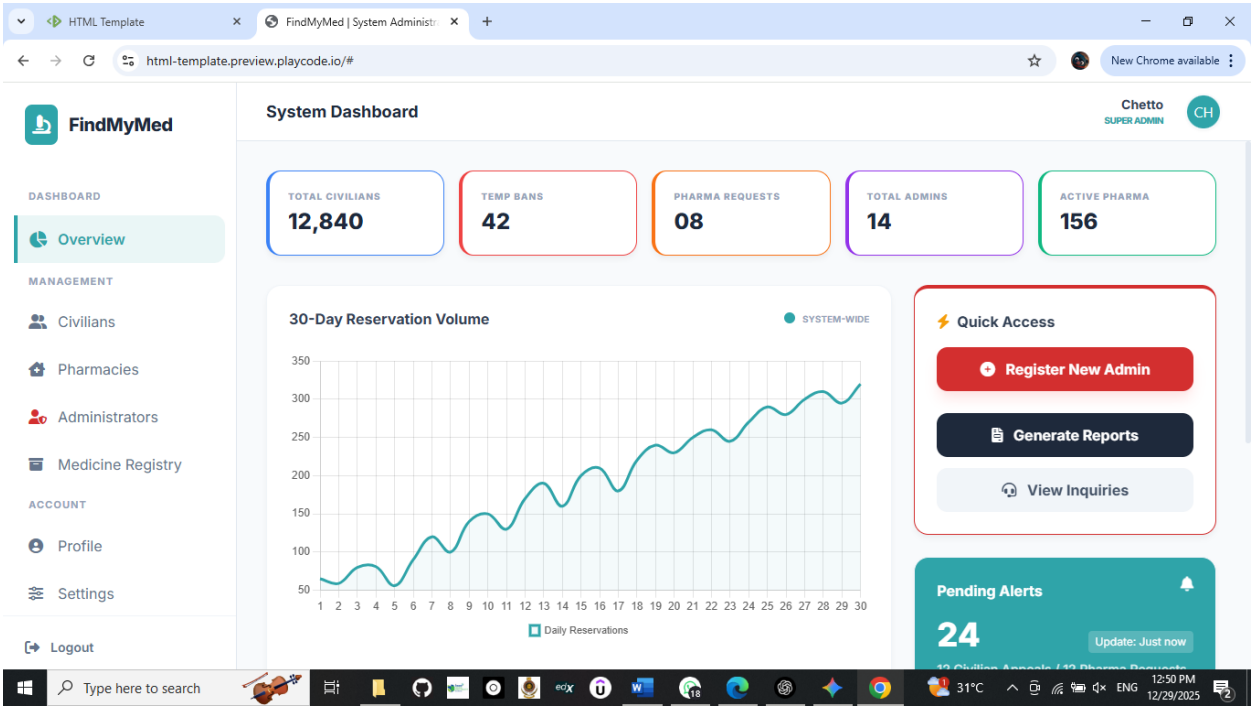
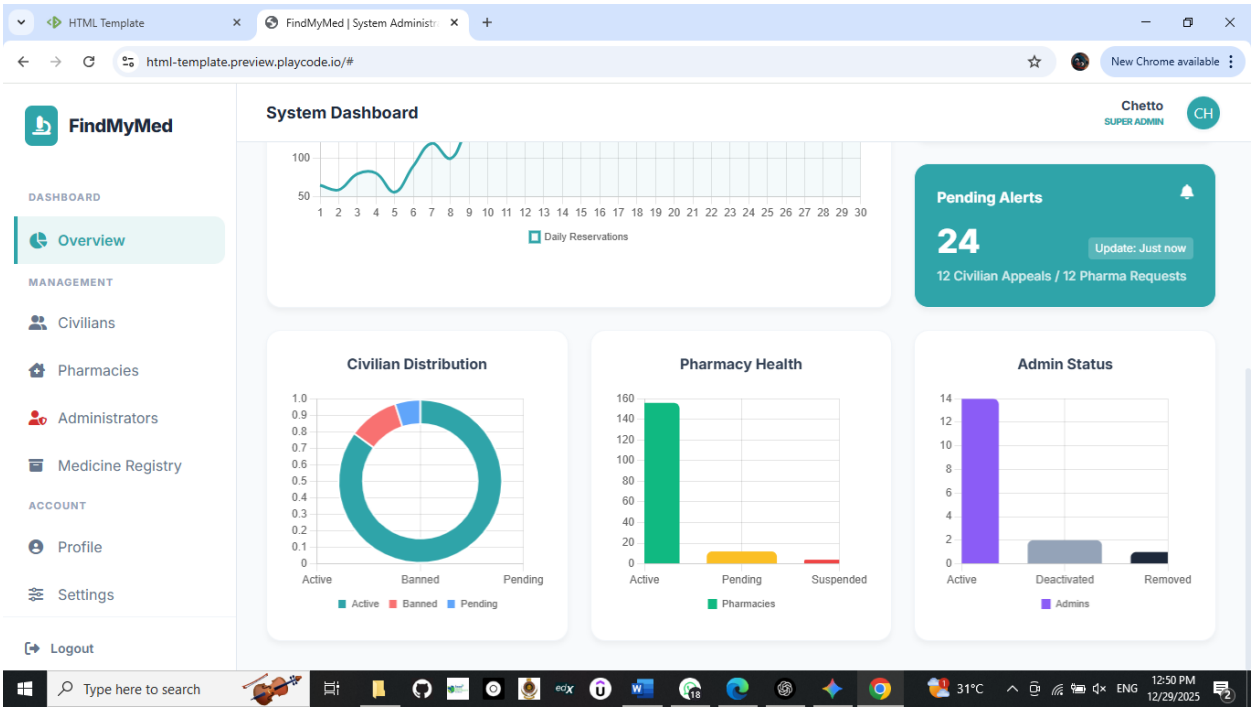
# Admin Panel Home Page






## 1. Left Sidebar (Navbar)

**Purpose:** Provides quick navigation to all major system modules.
**Placement:** Left side of the page, vertically aligned.
**Contents:**

- **Brand / Logo:** “FindMyMed” at the top.
- **Menu Items:**
  - Dashboard
  - Civilian Management
  - Pharmacy Management
  - Admin Management (**Super Admin only**)
  - Medicine Registry
  - Profile
  - Settings
  - Rules & Regulations
- **Behavior:**
  - Highlights the active module.
  - Collapsible on smaller screens (optional for responsiveness).
  - Role-based visibility: Admins cannot see Admin Management.

---

## 2. Top Section – Metric Cards

**Purpose:** Gives a quick snapshot of the most important system stats at a glance.
**Placement:** At the top of the main content area, arranged in a responsive grid layout.

**Metric Cards – Role-Based:**

**For Admin:**

- Total Civilians
- Temporary Bans
- Pending Civilian Appeals
- Pending Pharmacy Requests
- Total active pharmacies

**For Super Admin:**

- Total Civilians
- Total Admins
- Active Admins
- Total Pharmacies
- Pending Pharmacy Approvals

**Behavior:**

- Clicking a card highlights the related chart in the main section.
- Hovering may show more details, e.g., exact numbers or percentage change.
- Cards have color coding for quick visual identification (#2FA4A9 primary accent, complementary colors for variation).

---

## 3. Main Section – Visual Statistics

**Purpose:** Replace the old main table with interactive, visual analytics for easier understanding of system performance.

**Charts to Include:**

1. **Civilian Stats (Pie/Donut Chart):**
    - Distribution: Active / Temporarily Banned / Appeals Pending
    - Shows civilian management performance.
2. **Pharmacy Stats (Bar or Pie Chart):**
    - Pending Approvals
    - Active Pharmacies
    - Suspended Pharmacies
3. **Admin Stats (Bar Chart):**
    - Total Admins

    - Active Admins
    - Deactivated Admins
    - Removed Admins
4. **Reservation Analytics (Line Chart):**
    - Daily reservations over the last month
    - Helps track system usage trends

**Placement:**

- Charts arranged in a 2-column or 3-column responsive grid layout.
- Each chart has a small title, legend, and tooltip for details.

**Behavior:**

- Hovering over a chart displays exact numbers.
- Clicking a top metric card filters or highlights the relevant chart section.
- Interactive, dynamic, and responsive.

---

## 4. Right Sidebar – Quick Tools & Notifications

**Purpose:** Provides shortcuts for important actions and quick notification overview.

**Top Section (Tools):**

- Add Admin button (Super Admin only, red accent #d32f2f)
- Reports & Inquiry access button

**Bottom Section (Notification Metrics):**

- Read Notifications (primary accent #2FA4A9)
- Unread Notifications (primary accent #2FA4A9)

**Behavior:**

- Buttons open respective modules or forms.
- Clicking notification metrics could filter main charts or open the notification center.

---

## 5. Role-Based Behavior

**Super Admin:**

- Access to all metric cards and charts for civilians, pharmacies, admins, and system notifications.
- Tools in the right sidebar: Add Admin, Reports & Inquiries.

**Admin:**

- Sees metrics and charts only for civilians, pharmacies, and admin-related tasks.
- Restricted access: cannot see Admin Management.

**Dynamic Behavior:**

- Top metric cards filter or highlight related chart.
- Charts are interactive for deeper insights.

---

## 6. Summary of Components

| Component | Purpose | Contents | Behavior |
|---|---|---|---|
| Left Sidebar | Navigation | Logo, menu items (Dashboard, Civilian Management, etc.) | Active highlight, role-based visibility |
| Top Metric Cards | Quick stats | Total Civilians, Pending Requests, Admins, Pharmacies, Notifications | Click to filter charts, hover for details |
| Main Section | Visual analytics | Pie, Bar, Line charts | Interactive, responsive, hover tooltips, filtered by metric cards |
| Right Sidebar | Quick tools & notifications | Add Admin, Reports & Inquiry buttons, Read/Unread notification metrics | Buttons navigate to modules, notification metrics filter charts |
| Role-Based | Display adjustments | Admin vs Super Admin | Super Admin sees all, Admin sees restricted set |

---

## 7. Additional Notes

- **Removed Main Table:** Replaced with visual dashboards for clarity and better UX.
- **Responsive Design:** Layout adjusts for smaller screens: metric cards stack vertically, charts resize.
- **Color Scheme:** Primary accent #2FA4A9, red accent #d32f2f for Super Admin actions, white backgrounds, subtle shadows.

# Backend Implementation

## 1. Role-Based Access Control

- **Goal:** Super Admin sees all modules, Admin sees limited modules.
- **Implementation:**
    1. Add a `role` field in your `users` table (e.g., `ADMIN`, `SUPER_ADMIN`).
    2. When a user logs in, store their role in the session or JWT token.
    3. In the backend, fetch data conditionally based on role:

```
if(user.getRole().equals("SUPER_ADMIN")) {
    // Fetch all metrics and charts
} else if(user.getRole().equals("ADMIN")) {
    // Fetch limited metrics and charts
}
```

---

## 2. Metric Cards Data

- **Goal:** Display dynamic metrics like total civilians, temporary bans, pending appeals, etc.
- **Implementation:**

1. **Database queries for metrics:**

```
-- Total Civilians
SELECT COUNT(*) FROM civilians;

-- Temporary Bans
SELECT COUNT(*) FROM civilians WHERE status='TEMPORARY_BANNED';

-- Pending Civilian Appeals
SELECT COUNT(*) FROM civilian_appeals WHERE status='PENDING';

-- Pending Pharmacy Requests
SELECT COUNT(*) FROM pharmacy_requests WHERE status='PENDING';

-- Total Admins (Super Admin)
SELECT COUNT(*) FROM admins;

-- Active Admins
SELECT COUNT(*) FROM admins WHERE status='ACTIVE';

-- Total Pharmacies
SELECT COUNT(*) FROM pharmacies;

-- Pending Pharmacy Approvals
SELECT COUNT(*) FROM pharmacies WHERE approval_status='PENDING';

-- Admin Notifications
SELECT COUNT(*) FROM notifications WHERE type='ADMIN' AND user_id = :userId;
```

```
-- System Notifications
SELECT COUNT(*) FROM notifications WHERE type='SYSTEM';
```

2. **Backend Service Layer Example:**

```
@Service
public class DashboardService {
    @Autowired
    private JdbcTemplate jdbcTemplate;

    public Map<String, Integer> getMetrics(User user) {
        Map<String, Integer> metrics = new HashMap<>();
        metrics.put("totalCivilians", jdbcTemplate.queryForObject("SELECT
COUNT(*) FROM civilians", Integer.class));
        metrics.put("temporaryBans", jdbcTemplate.queryForObject("SELECT
COUNT(*) FROM civilians WHERE status='TEMPORARY_BANNED'", Integer.class));
        metrics.put("pendingAppeals", jdbcTemplate.queryForObject("SELECT
COUNT(*) FROM civilian_appeals WHERE status='PENDING'", Integer.class));

        if(user.getRole().equals("SUPER_ADMIN")) {
            metrics.put("totalAdmins", jdbcTemplate.queryForObject("SELECT
COUNT(*) FROM admins", Integer.class));
            metrics.put("activeAdmins", jdbcTemplate.queryForObject("SELECT
COUNT(*) FROM admins WHERE status='ACTIVE'", Integer.class));
            metrics.put("totalPharmacies",
jdbcTemplate.queryForObject("SELECT COUNT(*) FROM pharmacies",
Integer.class));
            metrics.put("pendingPharmacyApprovals",
jdbcTemplate.queryForObject("SELECT COUNT(*) FROM pharmacies WHERE
approval_status='PENDING'", Integer.class));
        }

        // Notifications
        metrics.put("adminNotifications", jdbcTemplate.queryForObject("SELECT
COUNT(*) FROM notifications WHERE type='ADMIN' AND user_id=?", Integer.class,
user.getId()));
        metrics.put("systemNotifications",
jdbcTemplate.queryForObject("SELECT COUNT(*) FROM notifications WHERE
type='SYSTEM'", Integer.class));

        return metrics;
    }
}
```

---

## 3. Charts Data

- **Goal:** Render charts dynamically based on metrics and historical data.
- **Example Queries:**

1. **Civilian Status Distribution (Pie Chart)**

```
SELECT status, COUNT(*) AS count FROM civilians GROUP BY status;
```

2. **Reservation Analytics (Line Chart)**

```
SELECT reservation_date, COUNT(*) AS total FROM reservations
WHERE reservation_date >= NOW() - INTERVAL 30 DAY
GROUP BY reservation_date;
```

3. **Pharmacy Stats**

```
SELECT approval_status, COUNT(*) AS count FROM pharmacies GROUP BY
approval_status;
```

**Backend Service:**

```
public Map<String, Object> getChartData(User user) {
    Map<String, Object> charts = new HashMap<>();
    charts.put("civilianStatus", jdbcTemplate.queryForList("SELECT status,
COUNT(*) AS count FROM civilians GROUP BY status"));
    charts.put("reservationAnalytics", jdbcTemplate.queryForList("SELECT
reservation_date, COUNT(*) AS total FROM reservations WHERE reservation_date
>= NOW() - INTERVAL 30 DAY GROUP BY reservation_date"));

    if(user.getRole().equals("SUPER_ADMIN")) {
        charts.put("pharmacyStats", jdbcTemplate.queryForList("SELECT
approval_status, COUNT(*) AS count FROM pharmacies GROUP BY
approval_status"));
        charts.put("adminStats", jdbcTemplate.queryForList("SELECT status,
COUNT(*) AS count FROM admins GROUP BY status"));
    }
    return charts;
}
```

---

## 4. Notifications

- **Goal:** Show Read/Unread notification metrics and quick notifications in right sidebar.
- **Implementation:**

1. **Queries:**

```
-- Unread notifications
SELECT COUNT(*) FROM notifications WHERE user_id=? AND is_read=0;

-- Read notifications
SELECT COUNT(*) FROM notifications WHERE user_id=? AND is_read=1;

-- Last 5 notifications for quick view
SELECT * FROM notifications WHERE user_id=? ORDER BY created_at DESC LIMIT 5;
```

2. **Backend Service Example:**

```
public Map<String, Object> getNotificationData(User user) {
    Map<String, Object> notifications = new HashMap<>();
```

```
    notifications.put("unreadCount", jdbcTemplate.queryForObject("SELECT
COUNT(*) FROM notifications WHERE user_id=? AND is_read=0", Integer.class,
user.getId()));
    notifications.put("readCount", jdbcTemplate.queryForObject("SELECT
COUNT(*) FROM notifications WHERE user_id=? AND is_read=1", Integer.class,
user.getId()));
    notifications.put("recentNotifications",
jdbcTemplate.queryForList("SELECT * FROM notifications WHERE user_id=? ORDER
BY created_at DESC LIMIT 5", user.getId()));
    return notifications;
}
```

---

## 5. Role-Based Data Filtering

- Metric cards and charts are filtered according to user role:
  - **Super Admin:** Full access to all metrics (admins, pharmacies, civilians).
  - **Admin:** Only sees civilians, pharmacies, and admin notifications relevant to their tasks.

---

## 6. Automatic Deletion Logic

- **Goal:** Remove old resolved/rejected notifications or reports automatically.
- **Implementation:**
  - Scheduled task using Spring Boot `@Scheduled`:

```
@Scheduled(cron = "0 0 0 * * ?") // Runs daily at midnight
public void deleteOldResolvedNotifications() {
    jdbcTemplate.update("DELETE FROM notifications WHERE is_read=1 AND
created_at <= NOW() - INTERVAL 7 DAY");
}

@Scheduled(cron = "0 0 0 * * ?")
public void deleteOldResolvedReports() {
    jdbcTemplate.update("DELETE FROM reports WHERE status IN ('RESOLVED',
'REJECTED') AND created_at <= NOW() - INTERVAL 60 DAY");
}
```

---

## 7. Controller Layer Example

```
@RestController
@RequestMapping("/api/dashboard")
public class DashboardController {
    @Autowired
    private DashboardService dashboardService;

    @GetMapping("/metrics")
    public ResponseEntity<Map<String,Integer>>
getMetrics(@AuthenticationPrincipal User user) {
        return ResponseEntity.ok(dashboardService.getMetrics(user));
    }
```

```
    @GetMapping("/charts")
    public ResponseEntity<Map<String,Object>>
getCharts(@AuthenticationPrincipal User user) {
        return ResponseEntity.ok(dashboardService.getChartData(user));
    }

    @GetMapping("/notifications")
    public ResponseEntity<Map<String,Object>>
getNotifications(@AuthenticationPrincipal User user) {
        return ResponseEntity.ok(dashboardService.getNotificationData(user));
    }
}
```

---

## ✅ Summary

1. **Role-Based Access:** Show different modules and metrics based on user role.
2. **Metric Cards:** Queries for counts dynamically; clicking cards filters charts.
3. **Charts:** Aggregate data for civilians, admins, pharmacies, reservations.
4. **Notifications:** Read/unread counts, recent notifications.
5. **Automatic Deletion:** Scheduled cleanup for old resolved/rejected items.
6. **Controller:** Provides JSON APIs for frontend to consume dynamically.