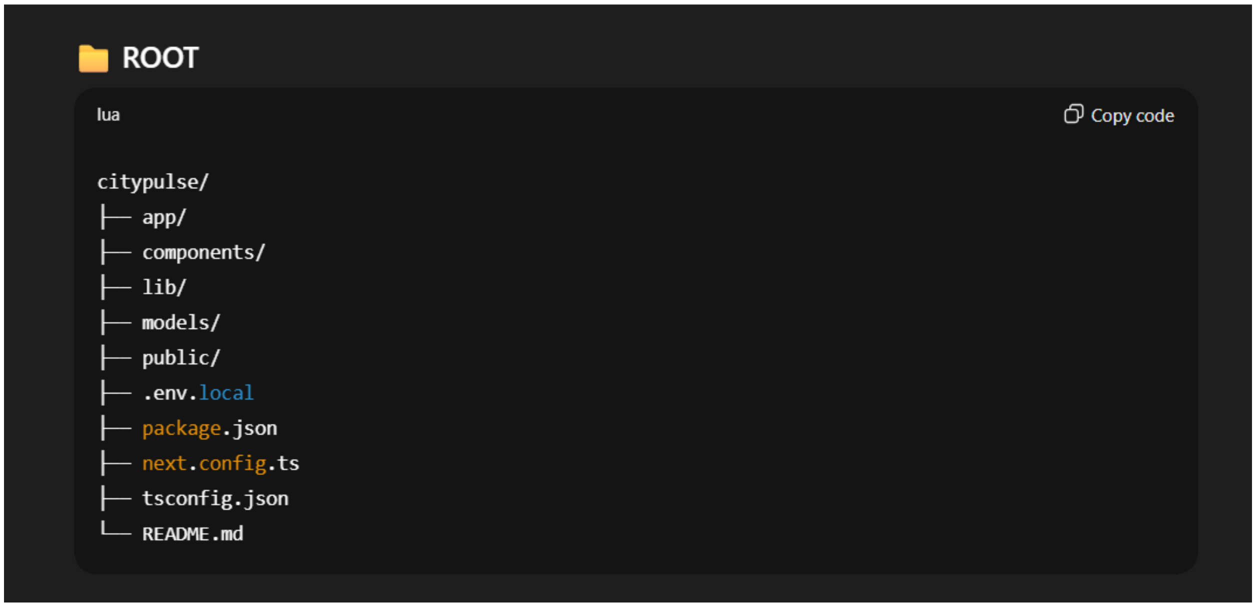
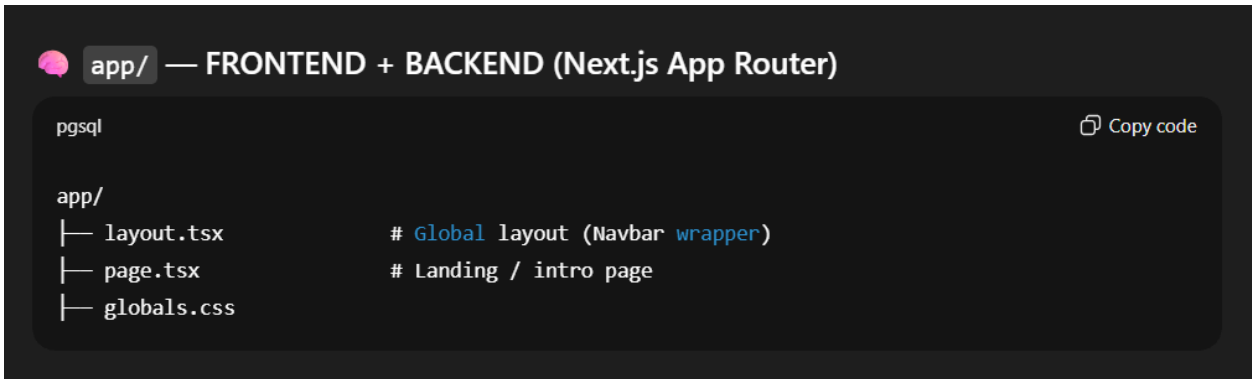


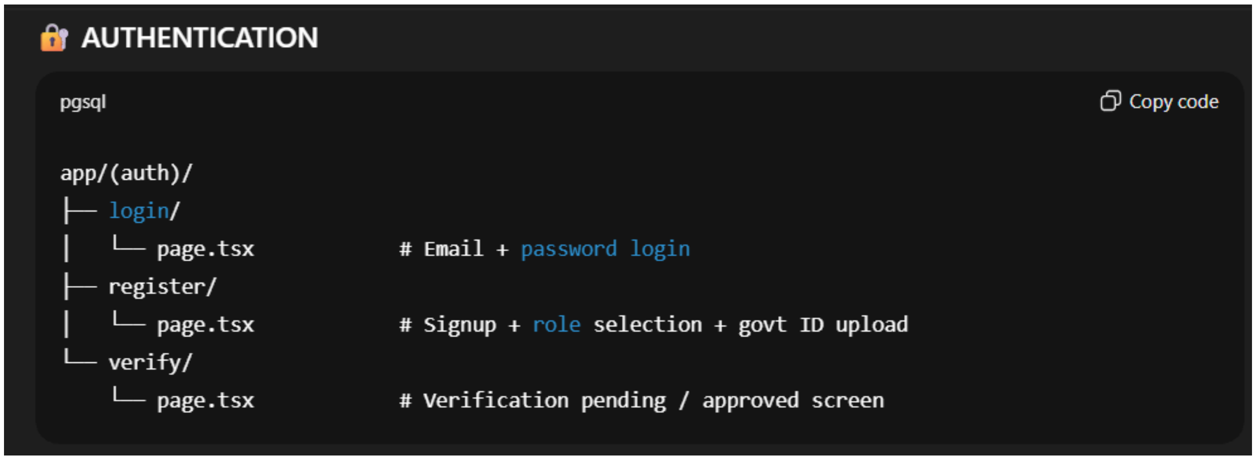
📁 ROOT

```
citypulse/
├── app/
├── components/
├── lib/
├── models/
├── public/
├── .env.local
├── package.json
├── next.config.ts
├── tsconfig.json
└── README.md
```



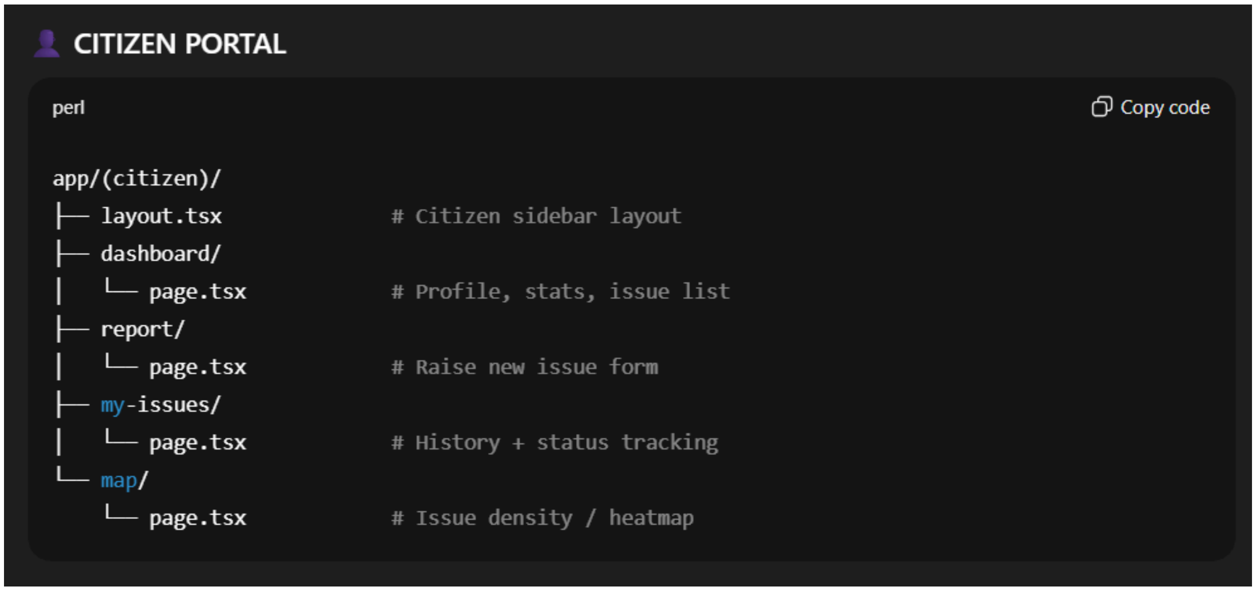
🧠 app/ — FRONTEND + BACKEND (Next.js App Router)

```
app/
├── layout.tsx              # Global layout (Navbar wrapper)
├── page.tsx                # Landing / intro page
├── globals.css
```



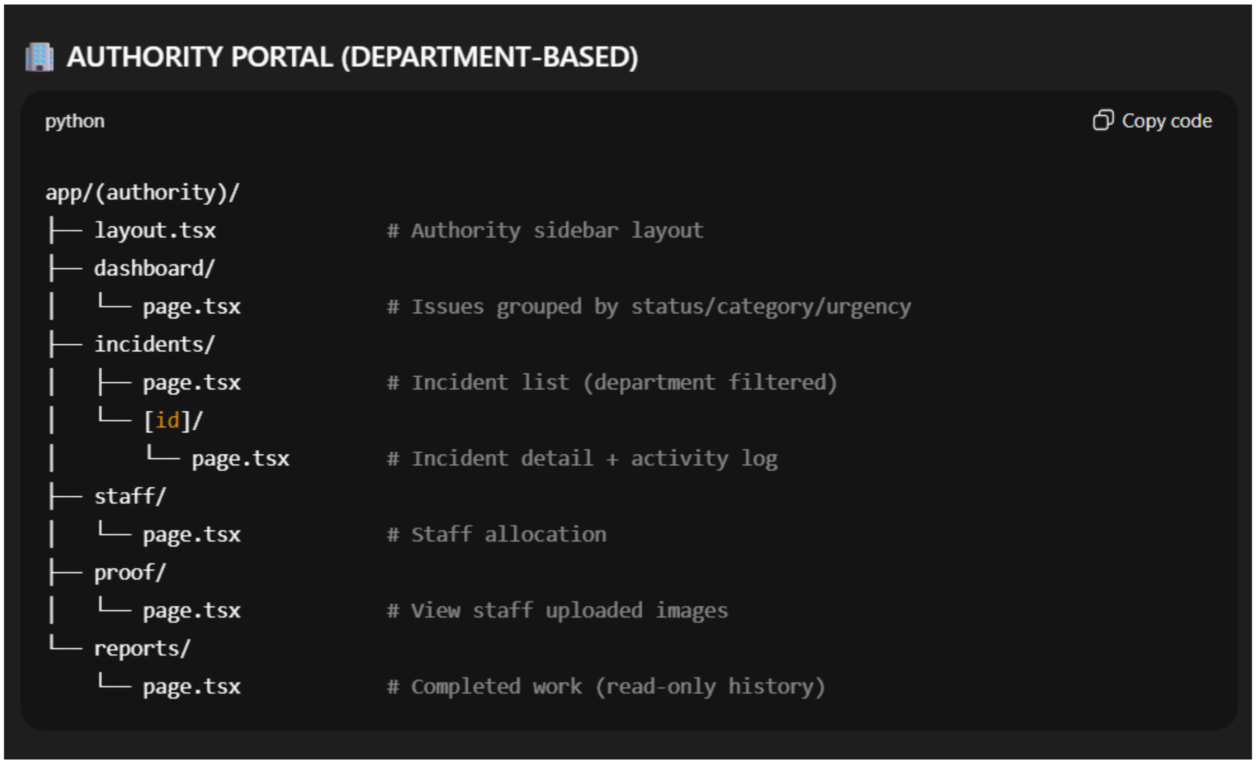
🔐 AUTHENTICATION

```
app/(auth)/
├── login/
│   └── page.tsx            # Email + password login
├── register/
│   └── page.tsx            # Signup + role selection + govt ID upload
└── verify/
    └── page.tsx            # Verification pending / approved screen
```



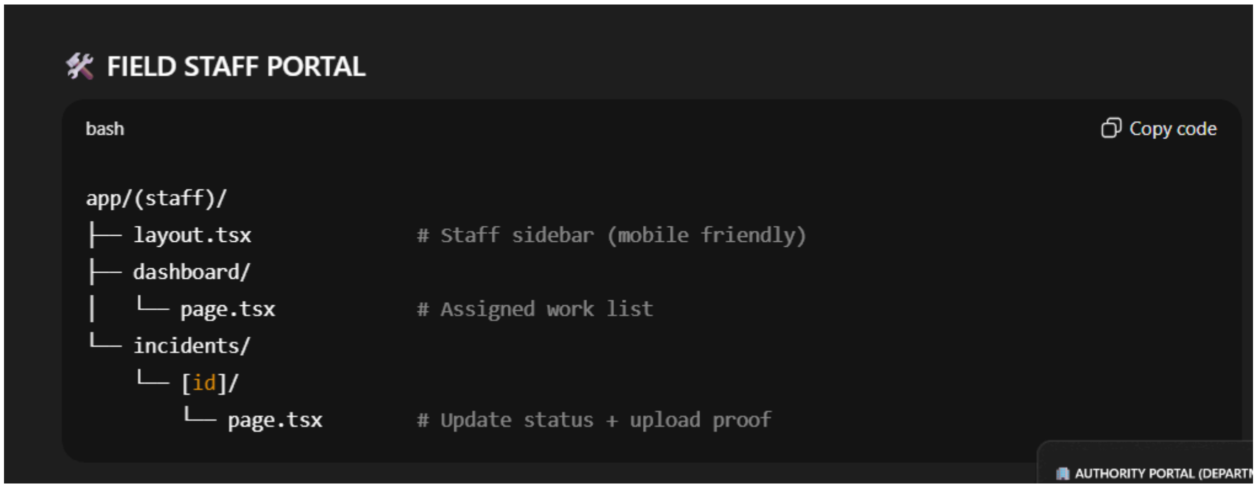
CITIZEN PORTAL

```
app/(citizen)/
├── layout.tsx              # Citizen sidebar layout
├── dashboard/
│   └── page.tsx            # Profile, stats, issue list
├── report/
│   └── page.tsx            # Raise new issue form
├── my-issues/
│   └── page.tsx            # History + status tracking
└── map/
    └── page.tsx            # Issue density / heatmap
```



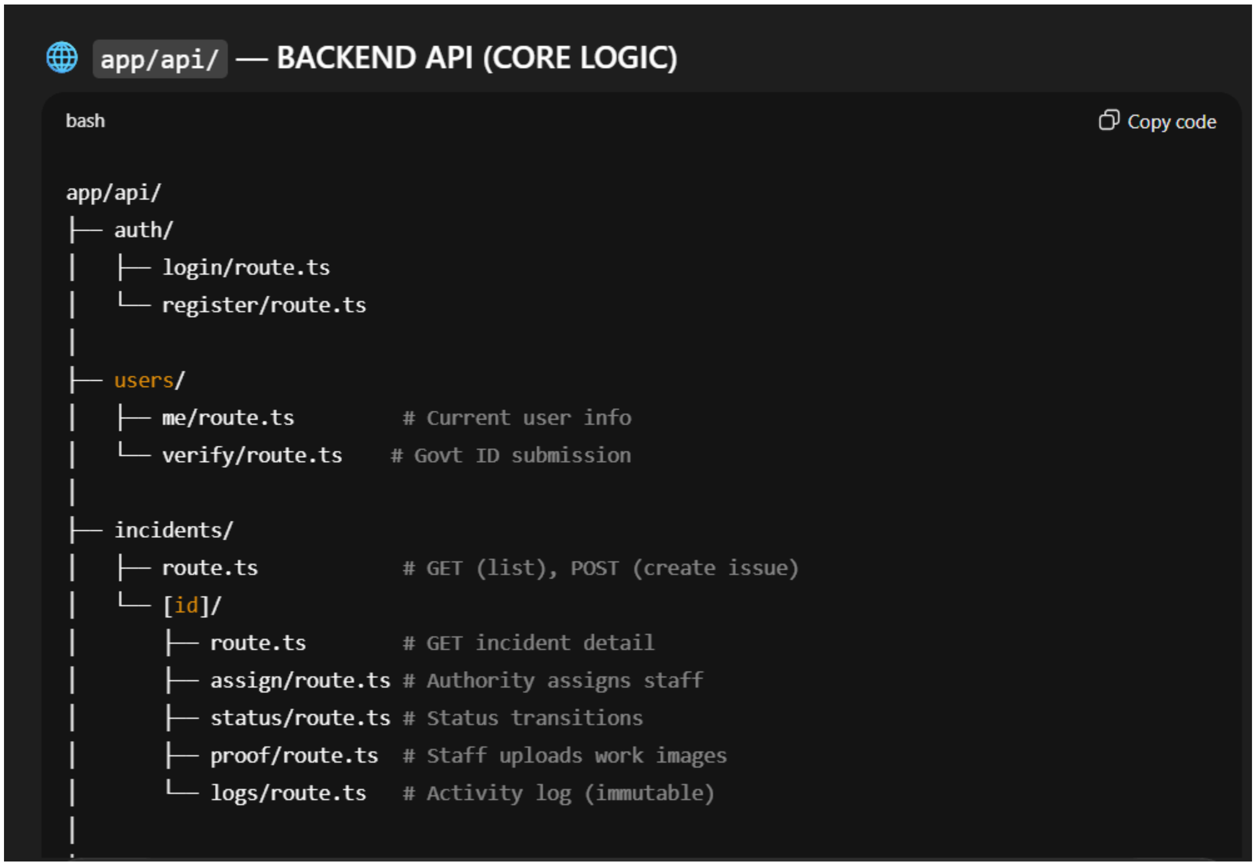
AUTHORITY PORTAL (DEPARTMENT-BASED)

```
app/(authority)/
├── layout.tsx              # Authority sidebar layout
├── dashboard/
│   └── page.tsx            # Issues grouped by status/category/urgency
├── incidents/
│   ├── page.tsx            # Incident list (department filtered)
│   └── [id]/
│       └── page.tsx        # Incident detail + activity log
├── staff/
│   └── page.tsx            # Staff allocation
├── proof/
│   └── page.tsx            # View staff uploaded images
└── reports/
    └── page.tsx            # Completed work (read-only history)
```



🛠 FIELD STAFF PORTAL

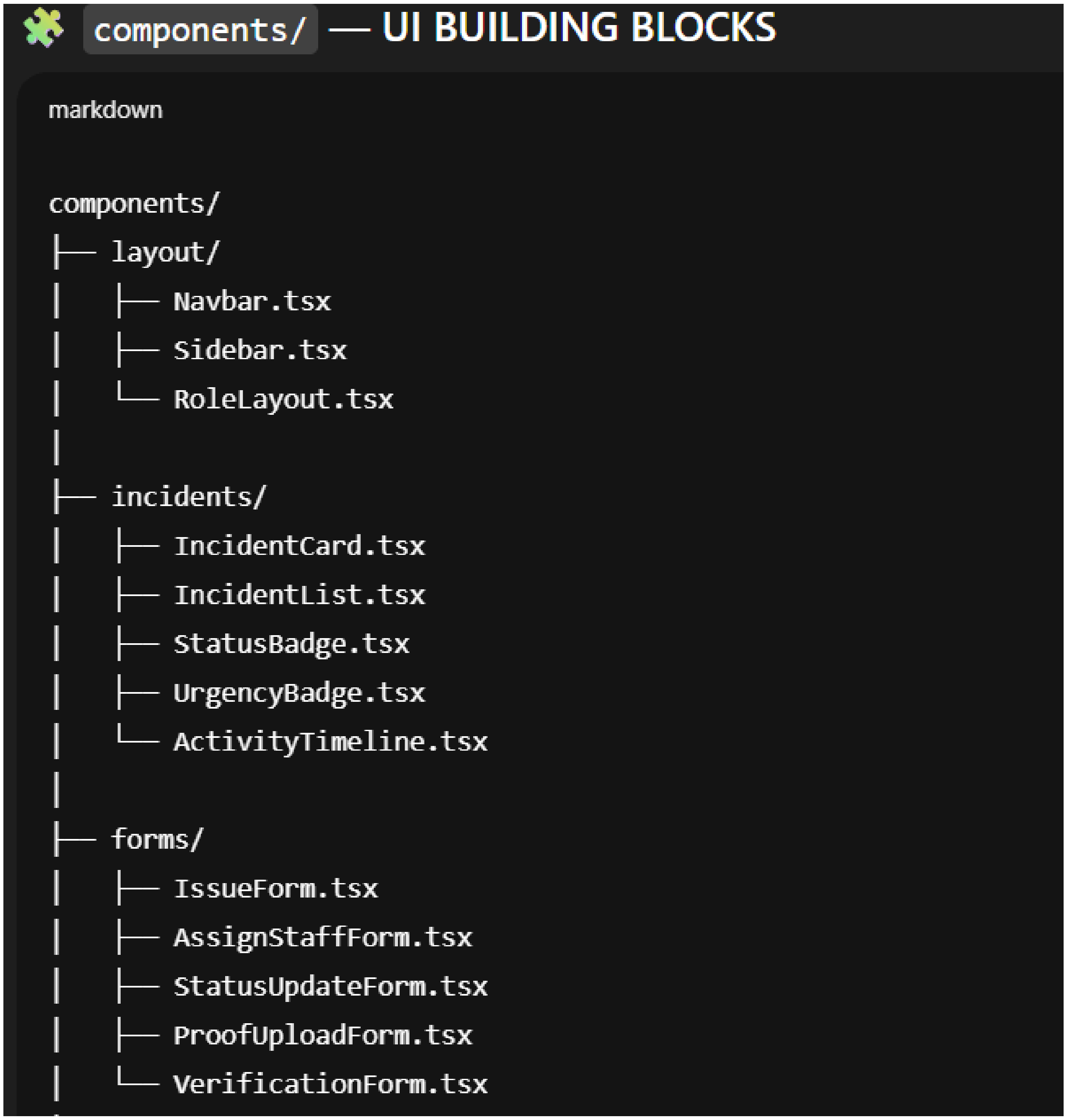
```bash
app/(staff)/
├── layout.tsx              # Staff sidebar (mobile friendly)
├── dashboard/
│   └── page.tsx            # Assigned work list
└── incidents/
    └── [id]/
        └── page.tsx        # Update status + upload proof
```



🌐 `app/api/` — BACKEND API (CORE LOGIC)

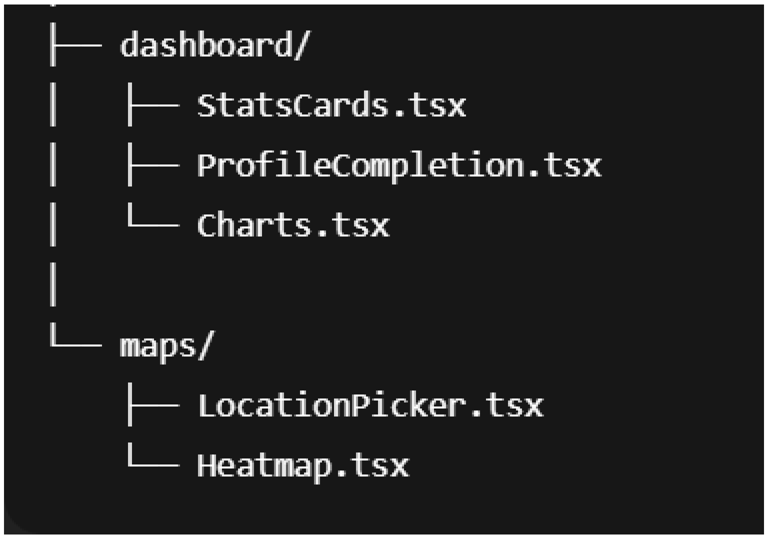
```bash
app/api/
├── auth/
│   ├── login/route.ts
│   └── register/route.ts
│
├── users/
│   ├── me/route.ts           # Current user info
│   └── verify/route.ts     # Govt ID submission
│
├── incidents/
│   ├── route.ts              # GET (list), POST (create issue)
│   └── [id]/
│       ├── route.ts          # GET incident detail
│       ├── assign/route.ts # Authority assigns staff
│       ├── status/route.ts # Status transitions
│       ├── proof/route.ts  # Staff uploads work images
│       └── logs/route.ts   # Activity log (immutable)
│
```

```
├── grouping/
│   └── route.ts           # Duplicate detection + frequency
│
├── metrics/
│   └── route.ts           # Stats for dashboards
│
└── public/
    └── incidents/route.ts  # Anonymized public data
```

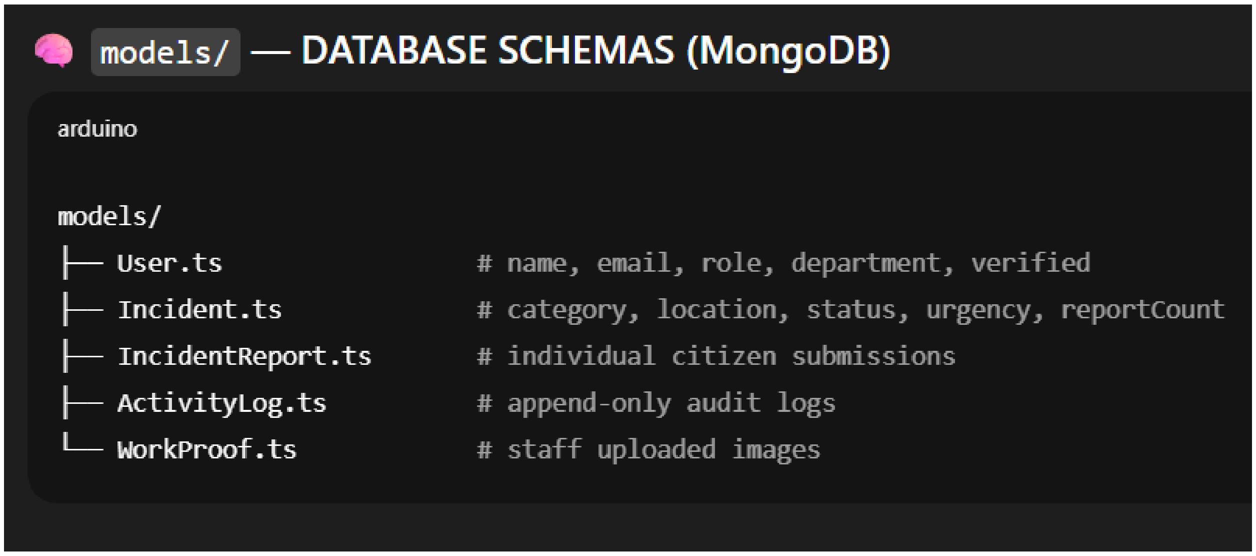

🧩 `components/` — UI BUILDING BLOCKS

markdown

```
components/
├── layout/
│   ├── Navbar.tsx
│   ├── Sidebar.tsx
│   └── RoleLayout.tsx
│
├── incidents/
│   ├── IncidentCard.tsx
│   ├── IncidentList.tsx
│   ├── StatusBadge.tsx
│   ├── UrgencyBadge.tsx
│   └── ActivityTimeline.tsx
│
├── forms/
│   ├── IssueForm.tsx
│   ├── AssignStaffForm.tsx
│   ├── StatusUpdateForm.tsx
│   ├── ProofUploadForm.tsx
│   └── VerificationForm.tsx
```



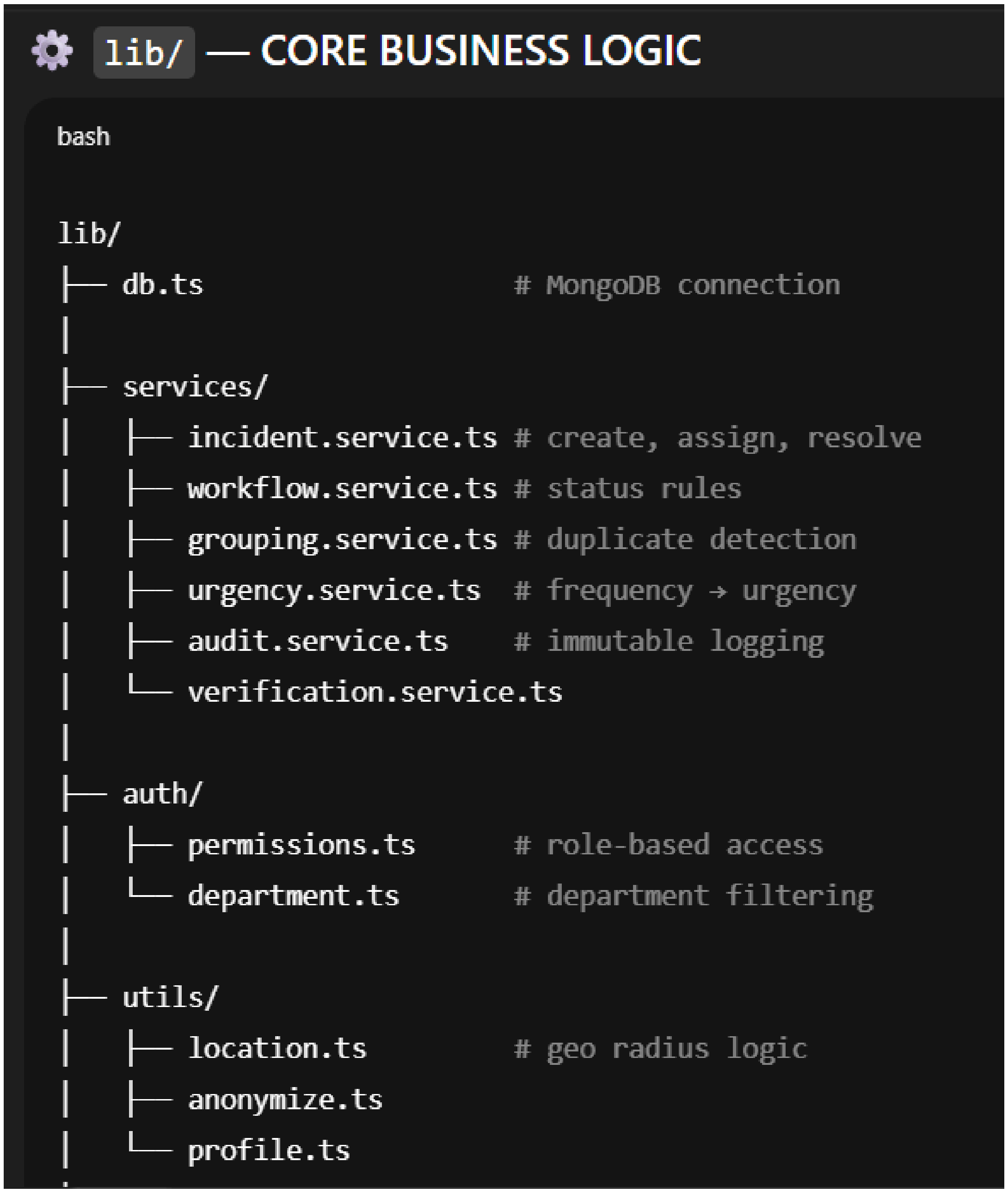
```
├── dashboard/
│   ├── StatsCards.tsx
│   ├── ProfileCompletion.tsx
│   └── Charts.tsx
│
└── maps/
    ├── LocationPicker.tsx
    └── Heatmap.tsx
```



🧠 `models/` — DATABASE SCHEMAS (MongoDB)

arduino

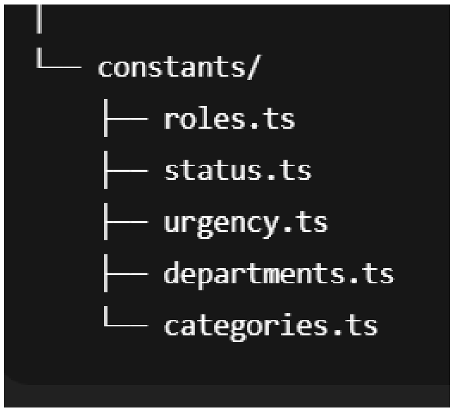
```
models/
├── User.ts              # name, email, role, department, verified
├── Incident.ts          # category, location, status, urgency, reportCount
├── IncidentReport.ts    # individual citizen submissions
├── ActivityLog.ts       # append-only audit logs
└── WorkProof.ts         # staff uploaded images
```



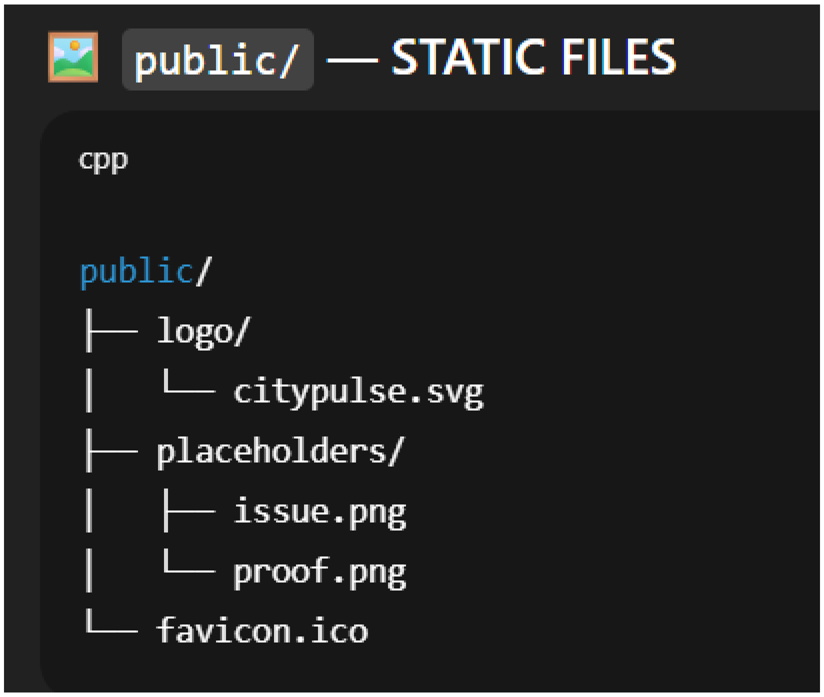
`lib/` — CORE BUSINESS LOGIC

bash

```bash
lib/
├── db.ts                    # MongoDB connection
│
├── services/
│   ├── incident.service.ts # create, assign, resolve
│   ├── workflow.service.ts # status rules
│   ├── grouping.service.ts # duplicate detection
│   ├── urgency.service.ts  # frequency → urgency
│   ├── audit.service.ts    # immutable logging
│   └── verification.service.ts
│
├── auth/
│   ├── permissions.ts      # role-based access
│   └── department.ts       # department filtering
│
├── utils/
│   ├── location.ts         # geo radius logic
│   ├── anonymize.ts
│   └── profile.ts
```



```bash
│
└── constants/
    ├── roles.ts
    ├── status.ts
    ├── urgency.ts
    ├── departments.ts
    └── categories.ts
```


🖼️ `public/` — STATIC FILES

cpp

```
public/
├── logo/
│   └── citypulse.svg
├── placeholders/
│   ├── issue.png
│   └── proof.png
└── favicon.ico
```

I need you to reorganize my Next.js project structure to match a specific architecture. Please reorganize the `app` and
api directories as follows.
Important:

1. Use Route Groups
2. (name) for organization.
3. For
4. (authority) and
5. (field-staff), you **MUST** create a sub-folder with the same name (or similar) inside the route group to avoid "parallel page" routing conflicts (e.g., `app/(authority)/authority/dashboard`).

Target Structure

App Directory (`app/`)

- **Root**: `layout.tsx`, `page.tsx`, `globals.css`
- **`(auth)`** (Authentication)
  - `/login/page.tsx`
  - `/register/page.tsx` (Note: rename from signup if necessary)
  - `/verify/page.tsx`
- **`(citizen)`** (Citizen Portal)
  - `layout.tsx` (Internal sidebar for citizens)
  - `/dashboard/page.tsx`
  - `/report/page.tsx`

    - `/my-reports/page.tsx`
    - `/map/page.tsx`
- `(authority)` (Authority Portal - **Namespaced**)
    - `layout.tsx` (Sidebar for authority)
    - **/authority/dashboard/page.tsx** (URL: `/authority/dashboard`)
    - **/authority/incidents/page.tsx**
    - **/authority/incidents/[id]/page.tsx**
    - **/authority/staff/page.tsx**
    - **/authority/proof/page.tsx**
    - **/authority/metrics/page.tsx**
- `(field-staff)` (Field Staff Portal - **Namespaced**)
    - `layout.tsx` (Mobile-friendly navbar)
    - **/field/assignments/page.tsx** (URL: `/field/assignments`)
    - **/field/incidents/[id]/page.tsx**

API Directory (`app/api/`)

- **`auth/`**: `login/route.ts`, `register/route.ts`, `verify-otp/route.ts`
- **`users/`**: `me/route.ts`, `verify/route.ts`
- **`incidents/`**: `route.ts`, `[id]/route.ts`, `[id]/status/route.ts`, `[id]/proof/route.ts`
- **`groups/`**: `route.ts`
- **`verification/`**: `route.ts`, `[id]/approve/route.ts`
- **`metrics/`**: `route.ts`
- **`public/`**: `incidents/route.ts`

Instructions

1. **Move existing files** to their best matching locations in this new structure.
2. **Create placeholder pages** for any missing routes (e.g., field staff pages, specific authority pages).
3. **Update imports** in any moved files to ensure relative paths (like `../../lib/...`) are correct.
4. **Update API calls** in your frontend pages to match the new API routes (e.g., `/api/auth/signup` -> `/api/auth/register`).
5. **Update `layout.tsx` files** in the new groups to link to the correct URLs (e.g., link to `/authority/dashboard` not just `/dashboard`).