# Backend One-Page (Django + DRF)

## Stack

1. Django
2. Django REST Framework
3. JWT (Simple JWT)
4. SQLite (current project setup)

Main folder: `/projectRoot/backend`

---

## Structure

Important apps:

1. `accounts` (auth/register/login/profile)
2. `rbac` (roles, permissions, dynamic access)
3. `cases` (complaints, crime scene reports, case lifecycle)
4. `evidence` (all evidence categories)
5. `investigation` (board workflow, suspects, interrogation)
6. `judiciary` (trial and verdict)
7. `rewards` (tips and reward flow)
8. `payments` (bail/fine payment optional flow)
9. `dashboard` (stats and module visibility)

---

## API Design

Base URL: `http://localhost:8000/api/`

Style:

1. RESTful endpoints
2. Role/permission checks per action
3. Domain-specific workflow endpoints (not only plain CRUD)

Auth:

1. JWT access/refresh tokens
2. Authenticated requests for protected APIs

---

## Data and Access Rules

1. Unique identity fields on users (username/email/phone/national ID).
2. RBAC is dynamic, role assignment editable by admin/superuser.
3. Multi-role users are supported.
4. Business rules enforced in serializers/views (case severity, workflow stage, authority checks).

---

## Payments (Optional Feature)

App: `/Users/reza/Documents/sag/backend/payments`

Covered:

1. Sergeant defines amount.
2. Eligible suspects/criminals based on severity and status.
3. Zarinpal sandbox request + verify flow.
4. Django template callback return page.

---

## Quality

1. API schema support via drf-spectacular.
2. Automated tests in multiple apps.
3. App-by-app separation for maintainability.

---

## Run Locally

Without Docker:

```
cd /projectRoot/backend
source .venv/bin/activate
python manage.py migrate
python manage.py seed_roles
python manage.py runserver
```

With Docker:

```
cd /projectRoot
docker compose up
```