Civic Issues App — Full Stack Scaffold

This document contains **copy-pasteable code** and instructions for a working MVP: React Native (Expo) mobile app, Node.js + Express backend with PostgreSQL, and a React (Vite) admin dashboard. Follow steps in order.

Project layout (suggested)

```
civic-project/
├─ backend/
│  ├─ package.json
│  ├─ server.js
│  ├─ db.js
│  ├─ routes/
│  │  ├─ auth.js
│  │  └─ complaints.js
│  ├─ uploads/        # local uploads for dev
│  ├─ Dockerfile
│  └─ docker-compose.yml

├─ mobile/            # Expo React Native
│  ├─ package.json
│  ├─ App.js
│  └─ screens/
│     ├─ LoginScreen.js
│     └─ ComplaintForm.js

└─ admin/             # Vite React admin dashboard
   ├─ package.json
   ├─ index.html
   ├─ src/
   │  ├─ main.jsx
   │  ├─ App.jsx
   │  └─ components/
   │     └─ ComplaintsTable.jsx
   └─ vite.config.js
```

---

1) Environment variables (backend `.env`)

```
PORT=4000
DATABASE_URL=postgres://postgres:postgres@db:5432/civic
JWT_SECRET=supersecretkey
FIREBASE_API_KEY=...             # optional if using Firebase
```

```
FIREBASE_AUTH_DOMAIN=...
FIREBASE_PROJECT_ID=...
FIREBASE_STORAGE_BUCKET=...
```

Replace values with your own.

---

2) Database schema (Postgres SQL)

```
-- Run this in psql or pgAdmin
CREATE DATABASE civic;
\c civic;

CREATE TABLE users (
  id SERIAL PRIMARY KEY,
  name VARCHAR(150),
  email VARCHAR(255) UNIQUE,
  phone VARCHAR(20) UNIQUE,
  created_at TIMESTAMP DEFAULT now()
);

CREATE TABLE admins (
  id SERIAL PRIMARY KEY,
  name VARCHAR(150),
  email VARCHAR(255) UNIQUE,
  password_hash VARCHAR(255),
  role VARCHAR(50) DEFAULT 'admin',
  created_at TIMESTAMP DEFAULT now()
);

CREATE TABLE complaints (
  id SERIAL PRIMARY KEY,
  user_id INTEGER REFERENCES users(id) ON DELETE SET NULL,
  category VARCHAR(100),
  description TEXT,
  image_url TEXT,
  latitude NUMERIC(9,6),
  longitude NUMERIC(9,6),
  status VARCHAR(20) DEFAULT 'pending',
  created_at TIMESTAMP DEFAULT now()
);

CREATE TABLE status_updates (
  id SERIAL PRIMARY KEY,
  complaint_id INTEGER REFERENCES complaints(id) ON DELETE CASCADE,
  admin_id INTEGER REFERENCES admins(id) ON DELETE SET NULL,
  status VARCHAR(20),
  comment TEXT,
  photo_url TEXT,
  updated_at TIMESTAMP DEFAULT now()
);
```

---

3) Backend — Node.js + Express

3.1 `backend/package.json`

```
{
  "name": "civic-backend",
  "version": "1.0.0",
  "main": "server.js",
  "scripts": {
    "start": "node server.js",
    "dev": "nodemon server.js"
  },
  "dependencies": {
    "cors": "^2.8.5",
    "dotenv": "^16.0.0",
    "express": "^4.18.2",
    "multer": "^1.4.5-lts.1",
    "pg": "^8.11.0",
    "bcrypt": "^5.1.0",
    "jsonwebtoken": "^9.0.0"
  },
  "devDependencies": {
    "nodemon": "^2.0.15"
  }
}
```

3.2 `backend/db.js`

```
// simple Postgres client using node-postgres
const { Pool } = require('pg');
require('dotenv').config();

const pool = new Pool({
  connectionString: process.env.DATABASE_URL,
});

module.exports = {
  query: (text, params) => pool.query(text, params),
  pool
};
```

3.3 `backend/server.js`

```
require('dotenv').config();
const express = require('express');
const cors = require('cors');
const app = express();
const authRoutes = require('./routes/auth');
const complaintRoutes = require('./routes/complaints');

app.use(cors());
app.use(express.json());
app.use('/uploads', express.static('uploads'));

app.use('/api/auth', authRoutes);
app.use('/api/complaints', complaintRoutes);

const PORT = process.env.PORT || 4000;
app.listen(PORT, () => console.log('Server listening on', PORT));
```

3.4 `backend/routes/auth.js`

```
const express = require('express');
const router = express.Router();
const db = require('../db');
const bcrypt = require('bcrypt');
const jwt = require('jsonwebtoken');

// Simple register (email) - for MVP. Replace with Firebase if desired.
router.post('/register', async (req, res) => {
  const { name, email, phone } = req.body;
  try {
    const result = await db.query(
      'INSERT INTO users(name, email, phone) VALUES($1,$2,$3) RETURNING *',
      [name, email, phone]
    );
    res.json({ user: result.rows[0] });
  } catch (err) {
    console.error(err);
    res.status(500).json({ error: 'DB error' });
  }
});

// Simple admin login stub (email + password)
router.post('/admin/login', async (req, res) => {
  const { email, password } = req.body;
  try {
    const r = await db.query('SELECT * FROM admins WHERE email=$1', [email]);
    if (!r.rows.length) return res.status(401).json({ error: 'Invalid' });
    const admin = r.rows[0];
    const ok = await bcrypt.compare(password, admin.password_hash);
    if (!ok) return res.status(401).json({ error: 'Invalid' });
    const token = jwt.sign({ id: admin.id, role: admin.role },
process.env.JWT_SECRET);
    res.json({ token, admin: { id: admin.id, name: admin.name, email:
admin.email } });
  } catch (err) {
    console.error(err);
    res.status(500).json({ error: 'Server error' });
  }
});

module.exports = router;
```

3.5 `backend/routes/complaints.js`

```
const express = require('express');
const router = express.Router();
const multer = require('multer');
const path = require('path');
const db = require('../db');

// local storage for dev
const storage = multer.diskStorage({
  destination: function (req, file, cb) {
    cb(null, 'uploads/');
  },
  filename: function (req, file, cb) {
    const ext = path.extname(file.originalname);
    cb(null, Date.now() + ext);
  }
});
const upload = multer({ storage });

// Create complaint
router.post('/', upload.single('photo'), async (req, res) => {
  try {
    const { description, latitude, longitude, category, user_id } = req.body;
    const image_url = req.file ? `/uploads/${req.file.filename}` : null;
    const insert = await db.query(
      `INSERT INTO complaints(user_id, category, description, image_url,
latitude, longitude)
       VALUES($1,$2,$3,$4,$5,$6) RETURNING *`,
      [user_id || null, category, description, image_url, latitude || null,
longitude || null]
    );
    res.json({ complaint: insert.rows[0] });
  } catch (err) {
    console.error(err);
    res.status(500).json({ error: 'DB error' });
  }
});

// Get complaints by user
router.get('/', async (req, res) => {
  const { userId } = req.query;
  try {
    const q = userId
      ? await db.query('SELECT * FROM complaints WHERE user_id=$1 ORDER BY 
created_at DESC', [userId])
      : await db.query('SELECT * FROM complaints ORDER BY created_at DESC');
    res.json({ complaints: q.rows });
  } catch (err) {
    console.error(err);
    res.status(500).json({ error: 'DB error' });
  }
});

// Get single complaint + status updates
router.get('/:id', async (req, res) => {
  const { id } = req.params;
  try {
    const c = await db.query('SELECT * FROM complaints WHERE id=$1', [id]);
    const updates = await db.query('SELECT * FROM status_updates WHERE 
complaint_id=$1 ORDER BY updated_at', [id]);
    res.json({ complaint: c.rows[0], updates: updates.rows });
  } catch (err) {
    console.error(err);
    res.status(500).json({ error: 'DB error' });
  }
});

// Admin update status
router.patch('/:id/status', upload.single('photo'), async (req, res) => {
  const { id } = req.params;
  const { status, comment, admin_id } = req.body;
  const photo_url = req.file ? `/uploads/${req.file.filename}` : null;
  try {
    await db.query('UPDATE complaints SET status=$1 WHERE id=$2', [status, 
id]);
    await db.query(
      'INSERT INTO status_updates(complaint_id, admin_id, status, comment, 
photo_url) VALUES($1,$2,$3,$4,$5)',
      [id, admin_id || null, status, comment || null, photo_url]
    );
    res.json({ success: true });
  } catch (err) {
    console.error(err);
    res.status(500).json({ error: 'DB error' });
  }
});
```

```
// Stats: solved %, counts
router.get('/stats/overview', async (req, res) => {
  try {
    const total = await db.query('SELECT COUNT(*) FROM complaints');
    const solved = await db.query("SELECT COUNT(*) FROM complaints WHERE
status='completed'");
    res.json({ total: Number(total.rows[0].count), solved:
Number(solved.rows[0].count) });
  } catch (err) {
    console.error(err);
    res.status(500).json({ error: 'DB error' });
  }
});

module.exports = router;
```

---

4) Docker & docker-compose (development)

`backend/Dockerfile`

```
FROM node:18
WORKDIR /app
COPY package*.json ./
RUN npm install
COPY . .
EXPOSE 4000
CMD ["node", "server.js"]
```

`docker-compose.yml` **(project root or backend folder)**

```
version: '3.8'
services:
  db:
    image: postgres:15
    environment:
      POSTGRES_DB: civic
      POSTGRES_USER: postgres
      POSTGRES_PASSWORD: postgres
    volumes:
      - pgdata:/var/lib/postgresql/data
    ports:
      - 5432:5432
  backend:
    build: ./backend
    environment:
      DATABASE_URL: postgres://postgres:postgres@db:5432/civic
      JWT_SECRET: supersecretkey
    ports:
      - 4000:4000
    depends_on:
      - db
    volumes:
      - ./backend/uploads:/app/uploads
volumes:
  pgdata:
```

Run with `docker-compose up --build`.

---

5) Mobile app (Expo React Native)

5.1 `mobile/package.json` (important deps)

```json
{
  "name": "civic-mobile",
  "main": "node_modules/expo/AppEntry.js",
  "scripts": {
    "start": "expo start",
    "android": "expo start --android",
    "ios": "expo start --ios"
  },
  "dependencies": {
    "expo": "~48.0.0",
    "expo-image-picker": "~14.0.1",
    "expo-location": "~15.0.1",
    "react": "18.2.0",
    "react-native": "0.71.8",
    "axios": "^1.4.0"
  }
}
```

(Version numbers are examples — run `npx create-expo-app` for latest.)

5.2 `mobile/App.js`

```javascript
import React from 'react';
import { NavigationContainer } from '@react-navigation/native';
import { createNativeStackNavigator } from '@react-navigation/native-stack';
import LoginScreen from './screens/LoginScreen';
import ComplaintForm from './screens/ComplaintForm';

const Stack = createNativeStackNavigator();
```

```
export default function App() {
  return (
    <NavigationContainer>
      <Stack.Navigator>
        <Stack.Screen name="Login" component={LoginScreen} />
        <Stack.Screen name="NewComplaint" component={ComplaintForm}
options={{ title: 'File Complaint' }} />
      </Stack.Navigator>
    </NavigationContainer>
  );
}
```

5.3 `mobile/screens/LoginScreen.js` (simple register/login by email)

```
import React, { useState } from 'react';
import { View, TextInput, Button } from 'react-native';
import axios from 'axios';

export default function LoginScreen({ navigation }) {
  const [email, setEmail] = useState('');
  const [name, setName] = useState('');

  const register = async () => {
    try {
      const res = await axios.post('http://10.0.2.2:4000/api/auth/register',
{ name, email });
      // naive:after register go to complaint form with user id
      const user = res.data.user;
      navigation.navigate('NewComplaint', { userId: user.id });
    } catch (err) {
      console.error(err);
      alert('Error registering');
    }
  };

  return (
    <View style={{ padding: 20 }}>
      <TextInput placeholder="Name" value={name} onChangeText={setName}
style={{ marginBottom: 10 }} />
      <TextInput placeholder="Email" value={email} onChangeText={setEmail}
style={{ marginBottom: 10 }} />
      <Button title="Register / Continue" onPress={register} />
    </View>
  );
}
```

5.4 `mobile/screens/ComplaintForm.js`

```
import React, {useState} from 'react';
import { View, Button, TextInput, Image, Text } from 'react-native';
import * as ImagePicker from 'expo-image-picker';
import * as Location from 'expo-location';
import axios from 'axios';

export default function ComplaintForm({ route }) {
  const userId = route?.params?.userId || null;
  const [desc, setDesc] = useState('');
  const [image, setImage] = useState(null);
  const [loc, setLoc] = useState(null);
  const [category, setCategory] = useState('pothole');

  const pickImage = async () => {
    const res = await ImagePicker.launchImageLibraryAsync({mediaTypes:
ImagePicker.MediaTypeOptions.Images});
    if (!res.cancelled) setImage(res.uri);
  };

  const getLocation = async () => {
    const { status } = await Location.requestForegroundPermissionsAsync();
    if (status !== 'granted') return alert('Permission denied');
    const location = await Location.getCurrentPositionAsync({});
    setLoc(location.coords);
  };

  const submit = async () => {
    if (!image || !loc) return alert('Select image and allow location');
    const form = new FormData();
    form.append('photo', {
      uri: image,
      name: 'photo.jpg',
      type: 'image/jpeg'
    });
    form.append('description', desc);
    form.append('latitude', String(loc.latitude));
    form.append('longitude', String(loc.longitude));
    form.append('category', category);
    form.append('user_id', String(userId || '')); // optional

    try {
      await axios.post('http://10.0.2.2:4000/api/complaints', form, {
        headers: { 'Content-Type': 'multipart/form-data' }
      });
      alert('Complaint submitted');
      setDesc(''); setImage(null); setLoc(null);
    } catch (err) {
      console.error(err);
      alert('Submission failed');
    }
  };

  return (
    <View style={{ padding: 20 }}>
      <Button title="Pick photo" onPress={pickImage} />
      {image && <Image source={{uri: image}} style={{width:200,height:200}} /
>}
      <Button title="Get Location" onPress={getLocation} />
      {loc && <Text>Lat: {loc.latitude}, Lon: {loc.longitude}</Text>}
      <TextInput value={desc} onChangeText={setDesc} placeholder="Describe
issue" style={{ marginVertical: 10 }} />
      <TextInput value={category} onChangeText={setCategory}
placeholder="Category (pothole/garbage)" />
      <Button title="Submit Complaint" onPress={submit} />
    </View>
  );
}
```

Notes: For Expo on Android emulator use `10.0.2.2` to reach host machine. For real device use your machine's LAN IP.

---

6) Admin dashboard (Vite + React)

6.1 `admin/package.json` (minimal)

```
{
  "name": "civic-admin",
  "version": "1.0.0",
  "scripts": {
    "dev": "vite",
    "build": "vite build",
    "preview": "vite preview"
  },
  "dependencies": {
    "axios": "^1.4.0",
    "react": "18.2.0",
    "react-dom": "18.2.0"
  },
  "devDependencies": {
    "vite": "^5.0.0"
  }
}
```

6.2 `admin/src/main.jsx`

```
import React from 'react'
import { createRoot } from 'react-dom/client'
import App from './App'
import './styles.css'

createRoot(document.getElementById('root')).render(<App />)
```

6.3 `admin/src/App.jsx`

```
import React, {useEffect, useState} from 'react';
import axios from 'axios';
import ComplaintsTable from './components/ComplaintsTable';

export default function App(){
  const [complaints, setComplaints] = useState([]);

  useEffect(()=>{
    fetch();
  },[]);

  const fetch = async ()=>{
    try{
      const res = await axios.get('http://localhost:4000/api/complaints');
      setComplaints(res.data.complaints);
    }catch(err){ console.error(err); }
  }

  return (
    <div style={{ padding: 30 }}>
      <h1>Municipality Admin</h1>
      <ComplaintsTable complaints={complaints} onUpdated={fetch} />
    </div>
  );
}
```

6.4 `admin/src/components/ComplaintsTable.jsx`

```
import React from 'react';
import axios from 'axios';

export default function ComplaintsTable({ complaints, onUpdated }){
  const changeStatus = async (id, status) => {
    try{
      await axios.patch(`http://localhost:4000/api/complaints/${id}/status`,
{ status });
      onUpdated();
    }catch(err){ console.error(err); }
  }

  return (
    <table border="1" cellPadding="8" style={{ width: '100%',
borderCollapse: 'collapse' }}>
      <thead>
        <tr>
          <th>ID</th><th>User</th><th>Category</th><th>Description</
th><th>Image</th><th>Status</th><th>Actions</th>
        </tr>
      </thead>
      <tbody>
        {complaints.map(c => (
          <tr key={c.id}>
            <td>{c.id}</td>
            <td>{c.user_id}</td>
            <td>{c.category}</td>
            <td>{c.description}</td>
            <td>{c.image_url ? <a href={`http://localhost:4000${c.image_url}
`} target="_blank">View</a> : '—'}</td>
            <td>{c.status}</td>
            <td>
              <button onClick={()=>changeStatus(c.id,'in_progress')}>In
Progress</button>
              <button onClick={()=>changeStatus(c.id,'completed')}
>Completed</button>
            </td>
          </tr>
        ))}
      </tbody>
    </table>
  );
}
```

---

7) Quick local run instructions

1. Start Postgres & backend using Docker (recommended):

   ```
   docker-compose up --build
   ```

   After DB is ready, run migrations (SQL from section 2).

2. Start backend (if not docker):

```
cd backend
npm install
npm run dev
```

3. Start mobile (Expo):

```
cd mobile
npm install
npx expo start
```

4. Start admin dashboard:

```
cd admin
npm install
npm run dev
```

---

8) Next improvements and production notes

- Replace local `uploads/` with Firebase Storage or AWS S3. Backend should upload to storage and save the storage URL.
- Use Firebase Auth or JWT-based auth with refresh tokens for users. Protect admin endpoints.
- Add validation, rate limiting, input sanitation.
- Use signed URLs for image downloads if privacy required.
- Add background worker or scheduler for SLA metrics.

---

9) Troubleshooting

- If images don't load in admin, ensure URLs point to `http://<host>:4000/uploads/<file>` and server serves `uploads/` statically.
- On Android emulator, use `10.0.2.2` to reach host machine; on iOS simulator use `localhost`.
- If Postgres connection refused in Docker, check `docker-compose` service names and `DATABASE_URL`.

---

10) Want any single part expanded?

If you want, I can now: - Generate a complete `backend` repo ZIP (files filled) or - Generate the full React Native app code expanded with navigation & UI or - Build a production-ready backend with S3/Firebase integration and JWT auth.

Tell me which and I will produce those files next.