STEP BY STEP WORK PROCEDURE

STEP 1 — Prepare system (prereqs)

Make sure these are installed:

- Node.js (v16+ / v18 recommended) — node -v
- npm (comes w/ Node) — npm -v
- Python 3.8+ — python --version
- pip — pip --version
- MongoDB (local) or an Atlas cluster — ensure it's running
- Git (optional)
- A terminal (Windows PowerShell / Terminal / WSL)

STEP 2 — Create copies & open project

1. Open your project folder in VS Code: WASTE/ (root contains smart-waste-backend and smart-waste-frontend).
2. Open two terminal tabs (one for backend, one for frontend).

STEP 3 — Backend: choose which to run

Your repo contains both Node.js backend files (controllers/*.js, routes/*.js, server.js, package.json) **and** Python backend files (app.py, requirements.txt, waste_model.h5, waste_classifier.py). You can run either:

- **If your server is Node.js** → follow **3A**
- **If your server is Python/Flask** → follow **3B**
- You may run both, but normally you use one.

STEP 3A — Node.js backend (if you use server.js / controllers/routes)

1. Open terminal, go to backend:

cd path/to/WASTE/smart-waste-backend

2. Install packages:

npm install

3. Create .env in smart-waste-backend with example values:

MONGO_URI=mongodb://localhost:27017/smartwaste

PORT=5000

JWT_SECRET=your_jwt_secret

4. Start MongoDB (if local):

- Windows (installed as service) — it may already be running.
- Or use mongod in another terminal if installed locally.

5. Start backend:

npm start

or, if package.json has no start, run:

node server.js

or using nodemon for dev:

npx nodemon server.js

6. Confirm server started (look for console logs like Server running on port 5000 or Connected to MongoDB).

STEP 3B — Python backend (if you use app.py / model endpoints)

1. Open terminal and create venv:

cd path/to/WASTE/smart-waste-backend

python -m venv venv

Windows

venv\Scripts\activate

macOS/Linux

source venv/bin/activate

2. Install requirements:

pip install -r requirements.txt

3. Create .env file (or set environment vars). Example .env:

FLASK_APP=app.py

FLASK_ENV=development

MONGO_URI=mongodb://localhost:27017/smartwaste

SECRET_KEY=your_secret

PORT=5000

(If the Python backend uses a different DB or sqlite, update accordingly.)

4. Start MongoDB (if required) and then run app:

python app.py

or, if it's a flask app using flask run:

flask run --port 5000

5. Confirm you see Running on http://127.0.0.1:5000/ or similar.

STEP 4 — Verify backend API endpoints

Use Postman or curl to list routes or test a couple endpoints:

- Check basic server/health:

curl http://localhost:5000/

- Typical auth endpoints (adjust if yours differ):

Register

curl -X POST http://localhost:5000/api/auth/register -H "Content-Type: application/json" -d '{"name":"Test","email":"test@example.com","password":"pass123"}'

Login

curl -X POST http://localhost:5000/api/auth/login -H "Content-Type: application/json" -d '{"email":"test@example.com","password":"pass123"}'

- Check classifier route (example — adjust to your route):

curl http://localhost:5000/api/classifier/predict

If any route paths differ, open routes/*.js or app.py to see the exact routes.

STEP 5 — Frontend: install & run (React)

1. Open a new terminal:

cd path/to/WASTE/smart-waste-frontend

npm install

2. Create .env in the frontend folder if not present:

REACT_APP_API_URL=http://localhost:5000

3. Start the React dev server:

npm start

4. The frontend should open at http://localhost:3000/. If not, open manually.

STEP 6 — Test full flow in browser

Visit these pages (based on your frontend src/pages files shown):

- Landing / Cover: http://localhost:3000/ (CoverPage / Home)
- Register: http://localhost:3000/register (Register.jsx)
- Login: http://localhost:3000/login (Login.jsx)
- User Dashboard: will be reachable after login (UserDashboard.jsx)
- Waste Classifier page: http://localhost:3000/waste-classifier or where your route is (WasteClassifier.jsx)
- Admin login / AdminPage: http://localhost:3000/admin / AdminPage.jsx
- Issues, Rewards: http://localhost:3000/report-issue / http://localhost:3000/rewards (check exact routes in App.jsx / main.jsx)

What to verify:

- Register → check MongoDB users collection for the new user
- Login → returns token / redirects to dashboard
- Dashboard → displays user info
- Waste classifier → upload image, see prediction
- Admin page → admin login flows and admin features

STEP 7 — Waste classifier specifics (model .h5 & Python)

You have waste_model.h5, plus waste_classifier.py and train_waste_classifier.py.

If classifier endpoint exists in Python backend:

1. Ensure tensorflow / keras installed in requirements.txt.
2. The classifier endpoint probably accepts multipart/form-data with an image. Example curl to test (adjust route):

curl -X POST http://localhost:5000/api/classifier/predict -F "image=@C:/path/to/test.jpg"

3. If you want to manually run the classifier script:

in backend venv

python waste_classifier.py --image path/to/test.jpg

or:

python evaluate_model.py

Open the classifier file to see expected input names and endpoints.

STEP 8 — Running both frontend & backend together (concurrently)

Option A: open two terminals — one runs backend, the other runs frontend (recommended).

Option B: use root-level concurrently if you want a single command — create root package.json with script:

"scripts": {

"dev": "concurrently \"npm start --prefix smart-waste-backend\" \"npm start --prefix smart-waste-frontend\""

}

Then run:

npm install -g concurrently

npm run dev

STEP 9 — Training the model (optional)

If you want to retrain:

Ensure proper python env activated

cd smart-waste-backend

python train_waste_classifier.py

or run the training notebook/script you have

Training uses dataset/train and dataset/val folders (your screenshots show categories). Expect heavy compute/time.

STEP 10 — Troubleshooting (common problems & fixes)

- **CORS errors**: add CORS support in backend (Express: npm i cors then app.use(require('cors')()), Flask: pip install -U flask-cors then CORS(app)).
- **Cannot connect to MongoDB**: ensure MongoDB service is running and MONGO_URI matches.
- **Frontend can't reach backend**: ensure REACT_APP_API_URL port matches backend. Check browser console for network calls.
- **Model load errors**: check TensorFlow / Keras versions and ensure waste_model.h5 path is correct.
- **Routes failing**: open the routes folder (authRoutes.js, classifierRoutes.js, etc.) to confirm exact paths and payloads.

STEP 11 — How to inspect code for exact endpoints & flow (quick)

- Backend (Node): open smart-waste-backend/routes/*.js to see endpoints (e.g., authRoutes.js).
- Backend (Python): open app.py and waste_classifier.py to see endpoints and payload formats.
- Frontend: open smart-waste-frontend/src/api.js or api.js (I saw one) and App.jsx / main.jsx to see routes used by frontend and the API base path.