from flask import Flask, render\_template, request, redirect, url\_for, flash, session

import config

from utils.data\_loader import ensure\_data\_files, load\_users

from utils.auth import create\_user, authenticate, find\_user\_by\_email, generate\_reset\_token, verify\_reset\_token, reset\_password, requires\_role, unlock\_account

from utils.viz import load\_minerals\_json

from datetime import timedelta

import os

app = Flask(\_\_name\_\_)

app.secret\_key = config.SECRET\_KEY

app.permanent\_session\_lifetime = config.PERMANENT\_SESSION\_LIFETIME

# ensure data files exist

ensure\_data\_files()

# Create initial admin if none exists (convenience for first run)

def create\_initial\_admin():

    users = load\_users().get("users", [])

    if not any(u["role"] == "Administrator" for u in users):

        # WARNING: in production, require env-admin or manual setup

        admin = create\_user("System", "Admin", "admin@example.com", "SouthAfrica", "MINN", "Administrator", "Admin@1234")

        print("Created default admin user:")

        print(" username:", admin["username"])

        print(" email: admin@example.com")

        print(" password: Admin@1234")

create\_initial\_admin()

# ----- Routes -----

@app.route("/")

def home():

    if "username" in session:

        role = session.get("role")

        if role == "Administrator":

            return redirect(url\_for("dashboard\_admin"))

        elif role == "Investor":

            return redirect(url\_for("dashboard\_investor"))

        else:

            return redirect(url\_for("dashboard\_researcher"))

    return redirect(url\_for("login"))

@app.route("/login", methods=["GET", "POST"])

def login():

    if request.method == "POST":

        username = request.form.get("username", "").strip()

        password = request.form.get("password", "")

        success, message = authenticate(username, password, remote\_ip=request.remote\_addr)

        if success:

            flash("Login successful.", "success")

            return redirect(url\_for("home"))

        else:

            flash(message, "danger")

    return render\_template("login.html")

@app.route("/logout")

def logout():

    session.clear()

    flash("Logged out.", "info")

    return redirect(url\_for("login"))

@app.route("/signup", methods=["GET", "POST"])

def signup():

    """

    Signup endpoint. You may restrict signup in production.

    Administrators should be created only by admin accounts ideally.

    """

    if request.method == "POST":

        first\_name = request.form.get("first\_name", "").strip()

        last\_name = request.form.get("last\_name", "").strip()

        email = request.form.get("email", "").strip()

        country = request.form.get("country", "").strip()

        org = request.form.get("organization", "").strip()

        role = request.form.get("role", "Researcher")

        password = request.form.get("password", "")

        confirm = request.form.get("confirm", "")

        # basic validations

        if not (first\_name and last\_name and email and country and password):

            flash("Please fill in all required fields.", "danger")

            return render\_template("signup.html")

        if password != confirm:

            flash("Passwords do not match.", "danger")

            return render\_template("signup.html")

        if find\_user\_by\_email(email):

            flash("An account with that email already exists.", "danger")

            return render\_template("signup.html")

        # prevent creating Administrator via public signup

        if role == "Administrator":

            flash("Administrator accounts must be created by an existing admin.", "danger")

            return render\_template("signup.html")

        new\_user = create\_user(first\_name, last\_name, email, country, org, role, password)

        flash(f"Account created. Your username is {new\_user['username']}. Please login.", "success")

        return redirect(url\_for("login"))

    return render\_template("signup.html")

# ----- Dashboard routes per role -----

@app.route("/dashboard/admin")

@requires\_role("Administrator")

def dashboard\_admin():

    users = load\_users().get("users", [])

    return render\_template("dashboard\_admin.html", users=users)

@app.route("/dashboard/investor")

@requires\_role("Investor")

def dashboard\_investor():

    return render\_template("dashboard\_investor.html")

@app.route("/dashboard/researcher")

@requires\_role("Researcher")

def dashboard\_researcher():

    minerals = load\_minerals\_json()

    return render\_template("dashboard\_researcher.html", minerals=minerals)

# ----- Password reset -----

@app.route("/reset-request", methods=["GET", "POST"])

def reset\_request():

    if request.method == "POST":

        email = request.form.get("email", "").strip()

        user = find\_user\_by\_email(email)

        if not user:

            flash("If an account exists with that email, a link will be sent.", "info")

            return redirect(url\_for("login"))

        token = generate\_reset\_token(email)

        # In production: send token via email. For this course: show link on screen (simulated)

        reset\_link = url\_for("reset\_password", token=token, \_external=True)

        flash("Password reset link generated (for demo it's shown below). In production send by email.", "info")

        return render\_template("reset\_request.html", reset\_link=reset\_link)

    return render\_template("reset\_request.html")

@app.route("/reset/<token>", methods=["GET", "POST"])

def reset\_password(token):

    email = verify\_reset\_token(token)

    if not email:

        flash("Invalid or expired token.", "danger")

        return redirect(url\_for("reset\_request"))

    if request.method == "POST":

        pw = request.form.get("password", "")

        confirm = request.form.get("confirm", "")

        if pw != confirm:

            flash("Passwords do not match.", "danger")

            return render\_template("reset\_password.html", token=token)

        reset\_password(email, pw)

        flash("Password has been reset. Please login.", "success")

        return redirect(url\_for("login"))

    return render\_template("reset\_password.html", token=token)

# ----- Admin unlock account action -----

@app.route("/admin/unlock/<user\_id>", methods=["POST"])

@requires\_role("Administrator")

def admin\_unlock(user\_id):

    ok = unlock\_account(user\_id)

    if ok:

        flash("Account unlocked.", "success")

    else:

        flash("Could not find user.", "danger")

    return redirect(url\_for("dashboard\_admin"))

# ---- Minerals Dashboard ----

from utils.viz import load\_minerals\_json, generate\_mineral\_chart, generate\_overview\_chart

@app.route("/minerals")

def minerals\_dashboard():

    """Display list of minerals with overview chart."""

    minerals = load\_minerals\_json()

    chart\_html = generate\_overview\_chart()

    return render\_template("minerals\_dashboard.html", minerals=minerals, chart\_html=chart\_html)

@app.route("/minerals/<mineral\_name>")

def mineral\_detail\_chart(mineral\_name):

    """Display individual mineral chart."""

    chart\_html = generate\_mineral\_chart(mineral\_name)

    return render\_template("minerals\_dashboard.html", chart\_html=chart\_html, single=True, mineral\_name=mineral\_name)

# Run

if \_\_name\_\_ == "\_\_main\_\_":

    app.run(debug=True)