from flask import Flask, request, jsonify, render\_template

from azure.storage.blob import BlobServiceClient

import os

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

# Azure Storage Configuration

AZURE\_STORAGE\_ACCOUNT\_NAME = "cloudfilesharingapp"

AZURE\_STORAGE\_ACCOUNT\_KEY = "YTGPC5ssauIcJ8bh2zo/ammuag2eNdXBjEQx2GGIyH8mw+WcBOrJawhuqXw58tIBEkfrtEWwRXqX+AStcrMvGw=="

AZURE\_CONTAINER\_NAME = "user-file"

# Connect to Azure Blob Storage

blob\_service\_client = BlobServiceClient(account\_url=f"https://{AZURE\_STORAGE\_ACCOUNT\_NAME}.blob.core.windows.net",

                                        credential=AZURE\_STORAGE\_ACCOUNT\_KEY)

@app.route("/")

def home():

    return render\_template("index.html")  # Serve frontend

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

def upload\_file():

    if "file" not in request.files:

        return jsonify({"error": "No file found!"}), 400

    file = request.files["file"]

    if file.filename == "":

        return jsonify({"error": "No selected file!"}), 400

    try:

        blob\_client = blob\_service\_client.get\_blob\_client(container=AZURE\_CONTAINER\_NAME, blob=file.filename)

        blob\_client.upload\_blob(file, overwrite=True)

        file\_url = f"https://{AZURE\_STORAGE\_ACCOUNT\_NAME}.blob.core.windows.net/{AZURE\_CONTAINER\_NAME}/{file.filename}"

        return jsonify({"message": "File uploaded successfully!", "file\_url": file\_url})

    except Exception as e:

        return jsonify({"error": str(e)}), 500

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

def list\_files():

    try:

        container\_client = blob\_service\_client.get\_container\_client(AZURE\_CONTAINER\_NAME)

        blob\_list = [f"https://{AZURE\_STORAGE\_ACCOUNT\_NAME}.blob.core.windows.net/{AZURE\_CONTAINER\_NAME}/{blob.name}" for blob in container\_client.list\_blobs()]

        return jsonify(blob\_list)

    except Exception as e:

        return jsonify({"error": str(e)}), 500

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

    app.run(debug=True)

ssS

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

from flask\_login import LoginManager, UserMixin, login\_user, login\_required, logout\_user, current\_user

from flask\_bcrypt import Bcrypt

import sqlite3

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

app.secret\_key = "supersecretkey"  # Change this in production

bcrypt = Bcrypt(app)

login\_manager = LoginManager()

login\_manager.init\_app(app)

# User class

class User(UserMixin):

    def \_\_init\_\_(self, id, username):

        self.id = id

        self.username = username

@login\_manager.user\_loader

def load\_user(user\_id):

    conn = sqlite3.connect("users.db")

    cursor = conn.cursor()

    cursor.execute("SELECT id, username FROM users WHERE id = ?", (user\_id,))

    user = cursor.fetchone()

    conn.close()

    if user:

        return User(user[0], user[1])

    return None

# ✅ Route: Register New User

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

def register():

    if request.method == "POST":

        username = request.form["username"]

        password = request.form["password"]

        hashed\_password = bcrypt.generate\_password\_hash(password).decode("utf-8")

        conn = sqlite3.connect("users.db")

        cursor = conn.cursor()

        try:

            cursor.execute("INSERT INTO users (username, password) VALUES (?, ?)", (username, hashed\_password))

            conn.commit()

            flash("✅ Registration successful! Please login.", "success")

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

        except:

            flash("❌ Username already exists!", "danger")

        finally:

            conn.close()

    return render\_template("register.html")

# ✅ Route: Login User

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

def login():

    if request.method == "POST":

        username = request.form["username"]

        password = request.form["password"]

        conn = sqlite3.connect("users.db")

        cursor = conn.cursor()

        cursor.execute("SELECT id, username, password FROM users WHERE username = ?", (username,))

        user = cursor.fetchone()

        conn.close()

        if user and bcrypt.check\_password\_hash(user[2], password):

            login\_user(User(user[0], user[1]))

            flash("✅ Login successful!", "success")

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

        else:

            flash("❌ Invalid credentials!", "danger")

    return render\_template("login.html")

# ✅ Route: Logout User

@app.route("/logout")

@login\_required

def logout():

    logout\_user()

    flash("✅ Logged out successfully!", "info")

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

# ✅ Route: Dashboard (Protected)

@app.route("/dashboard")

@login\_required

def dashboard():

    return render\_template("dashboard.html", username=current\_user.username)

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

    app.run(debug=True)