# import lib files

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

from datetime import datetime

import requests, json, re, random, math, pickle

import firebase\_admin

from firebase\_admin import db

# Initialize Firebase app

firebase\_admin.initialize\_app(options={'databaseURL': 'https://vitask.firebaseio.com/'})

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

# Change this to your secret key (can be anything, it's for extra protection)

app.secret\_key = 'canada$God7972#'

ref = db.reference('vitask')

slot\_a = {"x":50,"y":0,"price":15,"num":1}

slot\_b = {"x":0,"y":50,"price":20,"num":2}

slot\_c = {"x":50,"y":100,"price":25,"num":3}

slot\_d = {"x":100,"y":50, "price":30,"num":4}

# Homepage

@app.route('/')

def index():

    return render\_template('index.html')

# Dashboard

@app.route('/dashboard')

def dashboard():

# Check if user is loggedin

if 'loggedin' in session:

        if(session['parking\_space']=='A'):

            url = "https://thingspeak.com/channels/1208300/field/1.json"

        elif(session['parking\_space']=='B'):

            url = "https://thingspeak.com/channels/1208301/field/1.json"

        elif(session['parking\_space']=='C'):

            url = "https://thingspeak.com/channels/1208302/field/1.json"

        else:

            url = "https://thingspeak.com/channels/1208303/field/1.json"

        try:

            response = requests.get(url)

            data = json.loads(response.text)

            record = data['feeds'][-1]['field1']

            if record is None:

                record = 0

        except:

            record = 0

        slots = int(record)

        lastupdated\_nonformatted = datetime.now()

# dd/mm/YY H:M:S

lastupdated = lastupdated\_nonformatted.strftime("%d/%m/%Y %H:%M:%S")

        booked\_slots = []

        empty\_slots = []

        for i in range(0,slots):

            empty\_slots.append(i+1)

        for i in range(slots,10):

            booked\_slots.append(i+1)

        return render\_template('dashboard.html', username = session['username'], booked\_slots = booked\_slots, empty\_slots = empty\_slots, lastupdated = lastupdated, parking\_space = session['parking\_space'], distance = session['distance'], rate = session['rate'])

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

# Thingspeak write for parking spaces (1-4)

def reservedslots(parking\_space):

    temp = ref.child("parkfind").child("reserve").get()

    reservations = []

    if temp is not None:

        for key in temp:

            if(temp[key]["parkingSpace"]==parking\_space):

                reservations.append(temp[key])

    total = str(len(reservations))

# Select the parking space

if(parking\_space=='A'):

        url = "https://api.thingspeak.com/update?api\_key=7LHBQ6TZCKKWAYND&field1="+total

    elif(parking\_space=='B'):

        url = "https://api.thingspeak.com/update?api\_key=YBW96SSU1G299SSS&field1="+total

    elif(parking\_space=='C'):

        url = "https://api.thingspeak.com/update?api\_key=TOXRGRVZA1I28SIP&field1="+total

    else:

        url = "https://api.thingspeak.com/update?api\_key=I2L87XB3VT8P7L8V&field1="+total

    try:

        response = requests.get(url)

    except Exception as e:

        print(e)

# Distance Function

def calc\_distance(x1,y1,x2,y2):

    distance = math.sqrt(((x2-x1)\*\*2)+((y2-y1)\*\*2))

    return distance

# Reservation System

@app.route('/reservation')

def reservation():

    # Check if user is loggedin

    if 'loggedin' in session:

        # Check if account exists using Firebase

        reservations = []

        temp = ref.child("parkfind").child("reserve").get()

        if temp is not None:

            for key in temp:

                if(temp[key]["parkingSpace"]==session['parking\_space']):

                    reservations.append(temp[key])

        reservedslots(session['parking\_space'])

        return render\_template('reservation.html', username = session['username'], reservations = reservations)

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

# Reservation System

@app.route('/submit\_reservation', methods=['GET', 'POST'])

def submit\_reservation():

    # Check if user is loggedin

    if 'loggedin' in session:

        if request.method == 'POST' and 'carMark' in request.form and 'carNumber' in request.form:

            # Create variables for easy access

            carMark = request.form['carMark']

            carNumber = request.form['carNumber']

            parking\_space = session['parking\_space']

            username = session['username']

            # Add the reservation

            try:

                tut\_ref = ref.child("parkfind").child("reserve")

                tut\_ref.push({

                    'carMark': carMark,

                    'carNumber': carNumber,

                    'parkingSpace': parking\_space,

                    'username': username

                })

            except Exception as e:

                print(e)

        return redirect(url\_for('reservation'))

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

#Login

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

def login():

    # Output message if something goes wrong...

    msg = ''

    # Check if "username" and "password" POST requests exist (user submitted form)

    if request.method == 'POST' and 'username' in request.form and 'password' in request.form:

        # Create variables for easy access

        username = request.form['username']

        password = request.form['password']

        # Check if account exists using Firebase

        account = None

        temp = ref.child("parkfind").child("users").get()

        for key in temp:

            if(temp[key]["username"]==username):

                account = temp[key]

        # Fetch one record and return result

        # Login successful

        if(password==account["password"]):

            session['loggedin'] = True

            session['username'] = account["username"]

            # Generate Coordinates

            session['x'] = random.randint(0,100)

            session['y'] = random.randint(0,100)

            # Predict parking spot

            filename = 'finalized\_model.sav'

            distance\_a = calc\_distance(session['x'],session['y'],slot\_a['x'],slot\_a['y'])

            distance\_b = calc\_distance(session['x'],session['y'],slot\_b['x'],slot\_b['y'])

            distance\_c = calc\_distance(session['x'],session['y'],slot\_c['x'],slot\_c['y'])

            distance\_d = calc\_distance(session['x'],session['y'],slot\_d['x'],slot\_d['y'])

            data = [[distance\_a,distance\_b,distance\_c,distance\_d,15,20,25,30]]

            classifier = pickle.load(open(filename, 'rb'))

            parking\_space = classifier.predict(data)[0]

            all\_distance = {'A': round(distance\_a), 'B': round(distance\_b), 'C': round(distance\_c), 'D': round(distance\_d)}

            all\_rates = {'A': slot\_a['price'], 'B': slot\_b['price'], 'C': slot\_c['price'], 'D': slot\_d['price']}

            session['parking\_space'] = parking\_space

            session['distance'] = all\_distance[parking\_space]

            session['rate'] = all\_rates[parking\_space]

            # Redirect to dashboard

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

        else:

            # Account doesnt exist or username/password incorrect

            msg = 'Incorrect username/password!'

    # Show the login form with message (if any)

    return render\_template('login.html', msg=msg)

# Register

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

def register():

    # Output message if something goes wrong...

    msg = ''

    # Check if "username", "password" and "email" POST requests exist (user submitted form)

    if request.method == 'POST' and 'username' in request.form and 'password' in request.form:

        # Create variables for easy access

        username = request.form['username']

        password = request.form['password']

        # Check if account exists using Firebase

        snapshot = "Default"

        temp = ref.child("parkfind").child("users").get()

        for key in temp:

            if(temp[key]["username"]==username):

                snapshot = None

        if snapshot is None:

            account = True

        else:

            account = False

        # If account exists show error and validation checks

        if account:

            msg = 'Account already exists!'

        elif not re.match(r'[A-Za-z0-9]+', username):

            msg = 'Username must contain only characters and numbers!'

        elif not username or not password:

            msg = 'Please fill out the form!'

        else:

            # Account doesnt exists and the form data is valid, now insert new account into users table

            try:

                tut\_ref = ref.child("parkfind").child("users")

                tut\_ref.push({

                    'username': username,

                    'password': password

                })

                msg = 'You have successfully registered!'

            except Exception as e:

                print(e)

        print(msg)

    return render\_template('login.html', msg=msg)

# Web Logout

@app.route('/logout')

def logout():

    session.pop('username', None)

    session.pop('loggedin', None)

    session.pop('x', 0)

    session.pop('y', 0)

    session.pop('parking\_space', None)

    session.pop('distance', 0)

    session.pop('rate', 0)

    return redirect(url\_for('index'))

# Run the Flask Server

if \_\_name\_\_ == '\_\_main\_\_':

    app.run(debug=True)