```
from app import app, CURR_USER_KEY
from flask import request, render_template, flash, redirect, render_template, session, g, jsonify
from models import User, Story, Query
from helpers import *
from news_api_calls import *
from forms import RegisterForm, LoginForm, SearchForm
from sqlalchemy import exc
from psycopg2.errors import UniqueViolation



#TODONOW:
#-rewrite code and distinguish between session['saved] and session['results]


@app.before_request
def add_user_to_g():
    """If we're logged in, add curr user to Flask global."""
    if CURR_USER_KEY in session:
        g.user = User.query.get(session[CURR_USER_KEY])
    else:
        g.user = None


def do_login(user):
    """Log in user."""
    session[CURR_USER_KEY] = user.id


def do_logout(user):
    """Logout user."""
    if CURR_USER_KEY in session:
        del session[CURR_USER_KEY]



"""View functions for application"""


with app.app_context():
    @app.route('/')
    def homepage():
        # return redirect("/login")
        no_user = True
        if g.user:
            no_user = False
        """NewsTracker Homepage"""
        categories = ['business', 'entertainment',
                      'health', 'science', 'sports', 'technology']
        data = []
        results = async_reqs(categories)
        for index, result in enumerate(results):
            obj = {}
            obj['results'] = result
            obj['top_story'] = result[0]
            obj['name'] = categories[index].capitalize()
            data.append(obj)
        return render_template('/homepage.html', data=data, no_user=no_user)



@app.route('/headlines', methods=['GET', 'POST'])
def headlines():
    """Default Search Query if user logged in with a saved query as default"""
    if CURR_USER_KEY in session:
        user = User.query.get(g.user.id)
        queries = user.queries
        default = [query for query in queries if query.default]
        if default:
            return redirect(f"search/{default[0].id}")
        else:
            """Otherwise, Generic Top Headlines"""
            results = top_headlines_call()
            return render_template('/show_stories.html', results=results)
    results = top_headlines_call()
    return render_template('/show_stories.html', results=results)


@app.route(f'/headlines/<category>')
def show_for_category(category):
    """Display top headlines for given category based off link clicked from homepage"""
    category = category.lower()
    results = cat_calls(category, slideshow=False)
    return render_template('show_stories.html', results=results)


@app.route('/search', methods=['GET', 'POST'])
def search_form():
    """This function creates a dictionary extracting data from the search form to be sent to the news-
api"""
    if CURR_USER_KEY in session:
        form = SearchForm()
        if form.validate_on_submit():
            try:
                dict_query = form_query_to_dict(form)
                if form.saved_query.data or form.default.data:
```

```
                db_query = dict_query_to_db(g.user.id, session['query'])
                db.session.add(db_query)
                db.session.commit()
            advanced_search_call(dict_query)
            return redirect('/search/results')
        except:
            return render_template('/search.html', form=form)
    else:
        return render_template('/search.html', form=form)
  else:
    flash("You do not have permission to view this page. Please log-in and try again.", "danger")
    return redirect("/login")

@app.route('/search/<int:query_id>')
def search_user_queries(query_id):
    """Makes advanced search call based off of pre-saved query"""
    if CURR_USER_KEY in session:
        query_obj = Query.query.get(query_id)
        query_dict = db_query_to_dict(query_obj)
        advanced_search_call(query_dict)
        return redirect('/search/results')
    else:
        flash("You do not have permission to make this action. Please log-in and try again.", "danger")
        return redirect("/login")



@app.route('/search/results')
def handle_results():
    if CURR_USER_KEY in session:
        query = session['query']
        results = session['results']
        if query['sa'] == 'polarity':
            results = order_pol()
        elif query['sa'] == 'subjectivity':
            results = order_sub()
        else:
            return render_template('/show_stories.html', results=results)
    else:
        flash("You do not have permission to view this page. Please log-in and try again.", "danger")
        return redirect("/login")
    return render_template('/show_stories.html', results=results)


@app.route('/search/simple', methods=['GET'])
def search_simple():
    """API Call and Results for Simple Search"""
    keyword = request.args.get("search")
    results = simple_search_call(keyword)
    return render_template('/show_stories.html', results=results)


@app.route('/user/saved')
def user_saved():
    if CURR_USER_KEY in session:
        user = User.query.get(g.user.id)
        is_empty = False
        if len(user.saved_stories) == 0:
            is_empty = True
        if "saved" in session:
            session.pop("saved")
        session["saved"] = [story for story in user.saved_stories]
        return render_template("/user.html", user=user, is_empty=is_empty)
    else:
        flash("You do not have permission to view this page. Please log-in and try again.", "danger")
        return redirect("/login")



@app.route('/story/<id>/open')
def open_story_link(id):
    """Opens url associated with story when link is clicked"""
    try:
        story = Story.query.get(id)
        return redirect(f"{story.url}")
    except:
        results = session['results']
        story = [story for story in results if story['id'] == id][0]
        return redirect(f"{story['url']}")


@app.route('/story/<id>/save_story', methods=["POST"])
def save_story(id):
    if CURR_USER_KEY in session:
        results = session["results"]
        session_story = [story for story in results if story['id'] == id][0]
        story = Story(headline=session_story['headline'], source=session_story['source'],
content=session_story['content'],
                    author=session_story['author'], description=session_story['description'],
url=session_story['url'], image=session_story['image'],
                    published_at=session_story['published_at'], id=session_story['id'], user_id = g.user.id)
        try:
            db.session.add(story)
            db.session.commit()


        except exc.SQLAlchemyError as e:
            if isinstance(e.orig, UniqueViolation):
```

```
            flash(f'The story "{story.headline}" already exists in your saved stories.', 'danger')
            return redirect("/user/saved")
        return redirect("/user/saved")
    else:
        flash("You do not have permission to make this action. Please log-in and try again.", "danger")
        return redirect("/login")


@app.route('/story/<id>/remove', methods=["POST"])
def remove_story(id):
    if CURR_USER_KEY in session:
        Story.query.filter_by(id=id).delete()
        db.session.commit()
        return redirect("/user/saved")
    else:
        flash("You do not have permission to make this action. Please log-in and try again.", "danger")
        return redirect("/login")



@app.route('/register', methods=['GET', 'POST'])
def register_user():
    form = RegisterForm(prefix='form-register-')
    if form.validate_on_submit():
        username = form.username.data
        password = form.password.data
        email = form.email.data
        first_name = form.first_name.data
        last_name = form.last_name.data
        new_user = User.register(
            username, password, email, first_name, last_name)
        try:
            db.session.add(new_user)
            db.session.commit()
            flash("Congratulations! You have successfully created an account.", "success")
            do_login(new_user)
            return redirect('/headlines')
        except exc.SQLAlchemyError as e:
            if isinstance(e.orig, UniqueViolation):
                form.username.errors = [
                    "The username you entered is already taken. Please pick another one."]
            #
https://www.youtube.com/embed/iBYCoLhziX4?showinfo=0&controls=1&rel=0&autoplay=1
    else:
        print("Form Not Validated")
    return render_template('register.html', form=form)



@app.route('/login', methods=['GET', 'POST'])
def login_user():
    form = LoginForm()
    if form.validate_on_submit():
        username = form.username.data
        password = form.password.data
        user = User.authenticate(username, password)
        if user:
            do_login(user)
            flash("Credentials verified. You are now logged in.", "success")
            return redirect('/headlines')
        else:
            form.username.errors = [
                "Invalid username or password. Please try again."]
    return render_template('login.html', form=form)
```

```
@app.route('/login/demo', methods=['POST'])
def login_demo_user():
    random = uuid.uuid4().hex[:8]
    username = f"demo-user{random}"
    password = 'demouser'
    first_name = "first"
    last_name = "name"
    email = f"demo{random}@user.com"
    demo_user = User.register(
            username, password, email, first_name, last_name)
    db.session.commit()
    db.session.add(demo_user)
    db.session.commit()
    do_login(demo_user)
    return redirect('/headlines')


@app.route('/logout')
def logout():
    """Handle logout of user."""
    if CURR_USER_KEY in session:
        flash(f"You have successfully logged out.", "success")
        user = User.query.get(g.user.id)
        do_logout(user)
        return redirect("/headlines")
    else:
        return redirect("/login")

@app.route('/user/<int:query_id>/delete', methods=['POST'])
```

```
def remove_query(query_id):
   if CURR_USER_KEY in session:
      Query.query.filter_by(id=query_id).delete()
      db.session.commit()
      return jsonify({'response': f"Query deleted!"})
   else:
      flash("You do not have permission to make this action. Please log-in and try again.", "danger")
      return redirect("/login")
```