```python
from flask import Flask, session, redirect, url_for, escape, request, render_template, Blueprint, abort
import user
import sys
import bcrypt
import random
import debug_components
import datetime
from models import Observation, User, Comment, UnitProcess, Reference, \
    ComponentGroup, DataPointGroup, DataPoint, DataPointValue, \
    DataPointGroupNote
from models import db

def construct_blueprint(settings):
    blueprint = Blueprint(
        'observations',
        __name__,
        template_folder='templates')

    # This will return the template for the observations page. It has
    # been abstracted for re-use.
    def normal_observations_page():
        user = db.session.query(User) \
                         .filter_by(id=session['user_id']).one()
        observations = db.session.query(Observation).all()
        unit_processes = db.session.query(UnitProcess).all()
        return render_template(
            'observations.html',
            user = user,
            observations = observations,
            unit_processes = unit_processes,
            datetime=datetime
        )

    @blueprint.route('/observations', methods=['GET', 'POST'])
    def observations():
        if 'user_id' not in session:
            return redirect(url_for('logout.logout'))
        user = db.session.query(User) \
                         .filter_by(id=session['user_id']).one()

        if request.method == 'GET':
            return normal_observations_page()
        elif request.method == 'POST':
            form_component_id = request.form.get('create_unit_process_id')
            # form_component_id should be validated
            # ...

            if form_component_id is not None:
                try:
                    form_component_id = int(form_component_id)
                    # Get the form component with this ID, synthesize it,
                    # and add it to an observation, then add it to the stack
                    c = db.session.query(UnitProcess) \
                              .filter_by(id=form_component_id).one()
                    newobs = Observation(unit_process=c)
                    db.session.add(newobs)
                    db.session.commit()
                except:
                    db.session.rollback()
                return normal_observations_page()
            return 'post request not implemented', 500
        else:
            return 'Method not allowed', 405

    @blueprint.route('/observations/<int:obs_id>', methods=['GET'])
    def get_obs_id(obs_id):
        if 'user_id' not in session:
            return redirect(url_for('logout.logout'))
        user = db.session.query(User) \
                         .filter_by(id=session['user_id']).one()
        obs = db.session.query(Observation) \
                        .filter_by(id=obs_id).one()
        return render_template('observation.html', user=user,
                               observation=obs)

    @blueprint.route('/observations/<int:obs_id>/edit', methods=['POST'])
    def edit_obs_id(obs_id):
        if 'user_id' not in session:
            return redirect(url_for('logout.logout'))
        user = db.session.query(User) \
                         .filter_by(id=session['user_id']).one()
        observation = db.session.query(Observation) \
                                .filter_by(id=obs_id).one()
        dp_errors = {}
        return_anchor=None
        for key in request.form:
            value = request.form[key]
            if value == '':
                value = None
            if key.startswith('DataPoint_'):
                dp_id = int(key.split('_')[1])
                dpv = None

                # Attempt to locate a record for the DataPointValue
                try:
                    dpv = db.session.query(DataPointValue) \
                                    .filter_by(observation=observation,
                                               data_point_id=dp_id).one()
                except:
                    print >> sys.stderr, "Could not find DPV in database"
                    pass

                if dpv and dpv.value and value is None:
                    # Remove the DataPointValue
                    try:
                        db.session.delete(dpv)
                        db.session.commit()
                        continue
                    except:
                        print >> sys.stder, "Could not remove DataPointValue"
                        db.session.rollback()

                if value is not None:
                    try:
                        dpv = DataPointValue(observation=observation,
                                             data_point_id=dp_id,
                                             value=value)
                        db.session.add(dpv)
                        db.session.commit()
                    except AssertionError as e:
                        dp_errors[dp_id] = e.args[0]
                        db.session.rollback()
                    except Exception as e:
                        print >> sys.stderr, ("Could not write DataPointValue to database", e)
            if key.startswith('DataPointGroupNote_'):
                dpg_id = int(key.split('_')[1])
                try:
                    dpgn = db.session.query(DataPointGroupNote) \
                                     .filter_by(observation=observation,
                                                data_point_group_id=dpg_id) \
                                     .one()
                except:
                    dpgn = DataPointGroupNote(observation=observation,
                                              data_point_group_id=dpg_id,
                                              text=None)
                    db.session.add(dpgn)
                try:
                    if dpgn.text and dpgn.text != '' and value is None:
                        db.session.delete(dpgn)
                    else:
                        dpgn.text = value
                    db.session.commit()
                except:
                    db.session.rollback()
                    print >> sys.stderr, "Could not add DataPointGroupNote to observation."
            if key == 'return_anchor' and value and value != '':
                value = value.strip('#')
                return_anchor=value
        if dp_errors:
            return render_template('observation.html', user=user,
                                   observation=observation,
                                   dp_errors=dp_errors)
        return redirect(url_for('observations.get_obs_id',
                                obs_id=obs_id,
                                _anchor=return_anchor))


    @blueprint.route('/observations/<int:obs_id>/comment', methods=['POST'])
    def post_comment(obs_id):
        if 'user_id' not in session:
            return redirect(url_for('logout.logout'))
        user = db.session.query(User) \
                         .filter_by(id=session['user_id']).one()
        try:
            comment_text = request.form.get('comment_text')
            obs = db.session.query(Observation) \
                            .filter_by(id=obs_id).one()
            # Validate the comment text
            # ...
            new_comment = Comment(user=user, text=comment_text, \
                                  observation=obs)
            db.session.add(new_comment)
            db.session.commit()
        except:
            pass
        return redirect(url_for('observations.get_obs_id',
            obs_id=obs_id,
            _anchor='comments'))

    return blueprint

```