Assignment 3: Part 2

CS 496
Kyle Bedell
bedellk@oregonstate.edu

Tests explanation

The application lives here and does not require authentication:
https://project-4-144319.appspot.com/

Run the python script to list both line entries and users, then add a user and a line entry, and delete that user (which should also delete the user's associated line entry).

```
$ python tests.py
```

An explanation of the URL structure used to access resources

Get User IDs

GET /api/user

Get Line Entry IDs

GET /api/lineentry

Make New User

POST /api/user
parameters: first_name, last_name, email, password, user_type (admin or user)

Make new line entry

PUT /api/lineentry/user/[user id]

Delete user

DELETE /api/user/[user id]

A description of which RESTful constraints you met and which you did not (you do not need to make a RESTful app, but you do need to know why it does not meet all the constraints of a RESTful API)

This is a resource-based API which is largely not RESTful. It is based on the API presented in this week's final lecture.

It does not separate client and server, as it still has an interface powered by the server with Jinja2 templates in a web interface. It does not mention whether data is cacheable or not and for how long, and its messages do not mention MIME type. It also does not restrict its responses to requests that ask for JSON only.

In some way resources do include information on how to navigate to other resources, in the form of IDs to users and line entries. But this could be further expanded in future versions. It is stateless, in that it does not need to store any information about the client on it.

A discussion of any changes you needed to make to the schema from last week

I needed to add a to_dict() function for most entities that would return data needed for the API. Instead of returning the key in a urlsafe mode, I returned the numerical ID of the entity. I also converted datetime objects to strings for JSON parsing in these new functions.

Having finished the API, what you would have done differently

I regret the way I set up the inheritance of the entity types. Initially I thought it would be easier to distinguish between types of users, that is regular and administrators, by giving them distinct root ancestors.

This has turned out to be inefficient, as changes in one class of users may not be consistent quickly enough for the rest of the application. Therefore, you will notice in my test script that my DELETE test has to occur only after a few seconds wait.

I will refactor this oversight by giving the User class a knowledge of what kind of user it is.