Replenisher Task List

Environment:
Django 1.11
Python 3.4.1
Dateutil

Execute and test:

To create the database, run:
    python manage.py makemigrations
    python manage.py migrate

To run and test the webapp, run:
python manage.py runserver

Interface:
The App provide the following functionalities:

1. User related:
    — User Registration
    — User Login(Authorization)
    — Check individual User's profile and their tasks (the tasks assigned to the person or the person is the manager of the task)
    — Edit User profile

2. Task related:
    — Manager can create tasks and assign them to users who are managed by the manager
    — Users can create their own tasks for themselves.
    — Non-Manager user can't create tasks for other people.
    — Recurring tasks can be created and recurred by day/month/week/year, with or with out a due date of the whole set of tasks
    — Recurring tasks can be edit. The change can be applied to either individual task or all the recurring tasks.
    — Recurring tasks can be deleted by group (ie. delete all recurring tasks) or only delete one of the set of recurring tasks.
    — Each task can be edited or deleted by authorized user (ie. manager or assigned users) but can't be edited by the unauthorized user.

3. Feedback and Notes related:

— User can add general notes and feedback for tasks. Recurring tasks will share the general notes and feedbacks

— User can add unlimited number of notes and feedbacks to individual task by accessing each task's page. The note and feedbacks will only belongs to the individual task.

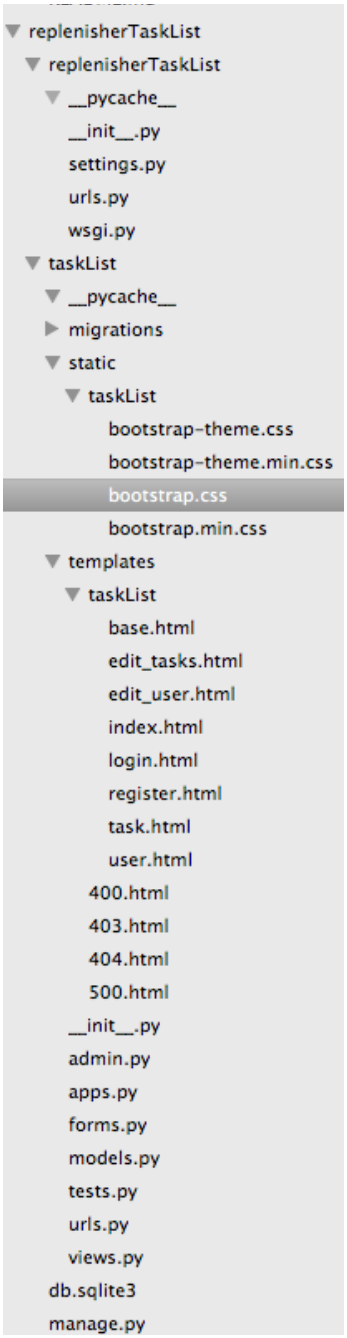
3. Admin related:

— Admin can build the relationship of management between users, so to define who can assign jobs to whom.

Directory structure:

The directory structure is as the picture.

- base.html: maintain the common elements across all the pages
- edit_tasks.html: the view to see when editing task
- edit_user.html: the view for editing user profile
- index.html: the default page contains tasks and to be used to add tasks
- login.html: login page
- register.html: registration page
- task.html : to display individual task
- user.html: to display each user's profile
- 400.html: use for 400 error, need to turn debug (in setting.py) off to see it
- 403.html: use for 403 error, need to turn debug (in setting.py) off to see it
- 404.html: use for 404 error, need to turn debug (in setting.py) off to see it
- 500.html: use for 500 error, need to turn debug (in setting.py) off to see it



Database Schema:

- See models.py for detail. The relationship is as following. There are several things worth to mention:

1. recurring is recorded in each task objects and also there will be one recurring task object to present each recurring task. The reason to do that is I want to allow users have freedom to edit both one task of a set of recurring task and all of the set of recurring task.

2. recurring task without repeat end date: when a recurring task without repeat end date is created, a certain number of tasks will be created (now I set it to 30). And everytime when the task list page is accessed, the number of recurring tasks will be checked and if there is some of them to be deleted, then the new tasks will be created since it's an recurring task.
3. each task has feedback and note column for general info (will be shared for all the task), and note and feedback objects for each of them.