Project Description

A Django application where authenticated users can create questionnaires that can be filled in by unauthenticated users. To create questionnaires you will use the built-in management system with additional changes (if necessary).

General considerations

- Collaborative development. You will use a solution of your choice (git, mercurial) and the trunk will be configured in a location accessible by team members
- The final configuration of the application for fast packaging and deployment (using a solution like fabric or pip) will be evaluated
- On delivery, the application shall be accompanied by the list of dependencies used in the project and a description of the installation process.

Definitions

- A **questionnaire** (or test) is defined by a name, a description, a fixed order collection of pages and a collection of possible outcomes.
- A **page** contains a fixed set of questions in a certain order.
- A **question** is composed of text and a collection of possible answers. Each possible answer is given a score (positive or negative). A question can have multiple answers.
- One **result** is a text associated with an upper limit and is showed to the user at the end of the test based on total points accumulated from responses.

Functionality

When a test is completed by a T, the final page displays a group of answers (along with corresponding questions) the user could have selected so that:
- the number of extra answers chosen is minimal
- following the selection of new responses the outcome would have been different (better or worse)
- in this algorithm must not be included two questions which belong to the same page.

The homepage of the application will list all the available tests (using pagination) including title, description, etc..

Bonus:
- Correctly addressing flows: It is intended that the user can not "jump" over pages by entering a URL manually (ex: you can not view a page that belongs to the test before filling in the answers to it) or skip some of the questions.
- Checking that the form is completed correctly is done for the whole page, not just for a single test.
- PEP8 code compliance

Project stages

A. Models & Admin

Documentation

- ✦ Python Tutorial http://docs.python.org/tutorial/
- ✦ The "first steps" in Django documentation

http://docs.djangoproject.com/en/1.1/#first-steps

- ✦ Section "The model layer" in the Django documentation

http://docs.djangoproject.com/en/1.1/#the-model-layer

- ✦ The "Admin Site" in the Django documentation

http://docs.djangoproject.com/en/1.1/ref/contrib/admin/#ref-contrib-admin

- ✦ The "Admin actions" in the Django documentation
  http://docs.djangoproject.com/en/1.1/ref/contrib/admin/actions/#ref-contrib-admin-actions
- ✦ Django Book, Chapters 1,2,5,6, http://www.djangobook.com/en/2.0/

Implementation

- ✦ Database diagram, component diagram
- ✦ Create project structure
- ✦ Creating Django models according to application needs
- ✦ Implementation of Django administration section for created models

B. Screens

Documentation

- ✦ Section "The template layer" in the Django documentation

http://docs.djangoproject.com/en/1.1/#the-template-layer especially the "For designers".

- ✦ Section "The view layer" in the Django documentation

http://docs.djangoproject.com/en/1.1/#the-view-layer especially "The Basics" and "Reference"

- ✦ The "Forms" in the Django documentation

http://docs.djangoproject.com/en/1.1/#forms

- ✦ Django Book, Chapters 3,4,7, http://www.djangobook.com/en/2.0/

Implementation

- ✦ URL map
- ✦ Templates (application's screens) without fully or properly implemented flows.

C. Final

Documentation

- ✦ The rest of Django documentation.
- ✦ The rest of Django Book.

Implementation

- ✦ Completing the *logic* part of the application (views).
- ✦ Documenting the code
- ✦ Unittesting