**CS4720 Summer 2019**

**Instructions:**

1. Detection of plagiarism will result in receiving the failing grade. Students must complete the final project by themselves.
2. The most recent versions of Python (3.6.x) and Django (2.0.4) have to be used for the final project.
3. Submission instruction: Students must submit a **report file** and a **zipped project folder**. In addition, the following files have to be converted into text files and be submitted individually (do not zip)
   1. Models.py
   2. Final.html (to display the “List of Areas” table)
4. Submission Example (only for Assignment 6):
   1. Assignment folder: Final Project-Codes (zipped file)
      1. Codes.zip
   2. Assignment folder: Final Project-Codes in Document and Report
      1. Models.txt
      2. Final.txt
      3. Reports.docx
5. Failure to follow the rules will result in deducting points.

**Final Project (100 points)**

In this final project, you will develop an application in **Django**. This application will include a model, some administrative interface customization and a view.

The application model is based on the measurements example as described in the “Final Project-Measurements Example” file.

Create a Django project with an application named measurements. This will be the base project for this assignment. You should name the project final so that there is no confusion between the project and the application.

**Model (35 points)**

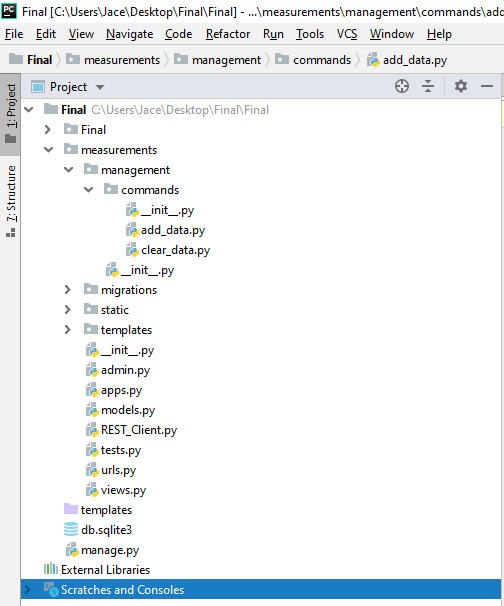
Some code will be provided to initialize data for this example, but you will need to be careful about field and class names. All classes described here are model classes, so they extend models.Model. Each class should define \_\_str\_\_ to return the name of the instance, except for Location and Measurement which are described below.

* Area (15 points)
  + id: integer, primary key
  + name: characters
  + longitude: float
  + latitude: float
  + Several helper methods (10 points)
    - number\_of\_locations which returns the number of locations for this area
    - average\_measurement which returns the average of the measurements for this area. If the area has no measurements then return None.
    - category\_names which returns a string with a list of categories that this area belongs to. The names should be comma separated. If the area belongs to no categories, return the empty string.
* Location (5 points)
  + id: integer, primary key
  + name: characters
  + altitude: integer
  + area: foreign key referencing Area
  + The \_\_str\_\_ method should return the concatanation of the name of the area the location belongs to, a colon and the name of the location. So, something like “Grand Canyon:North Rim”
* Measurement (5 points)
  + id: integer, primary key
  + value: float
  + date: a date-time field
  + location: A foreign key referencing Location
  + The \_\_str\_\_ method should return the concatanation of the string “measurement@” with the location string representation. So, something like “measurement@Grand Canyon:North Rim”
* Category (5 points)
  + id: integer, primary key
  + name: characters
  + description: characters
  + members: A many-many field referencing Area

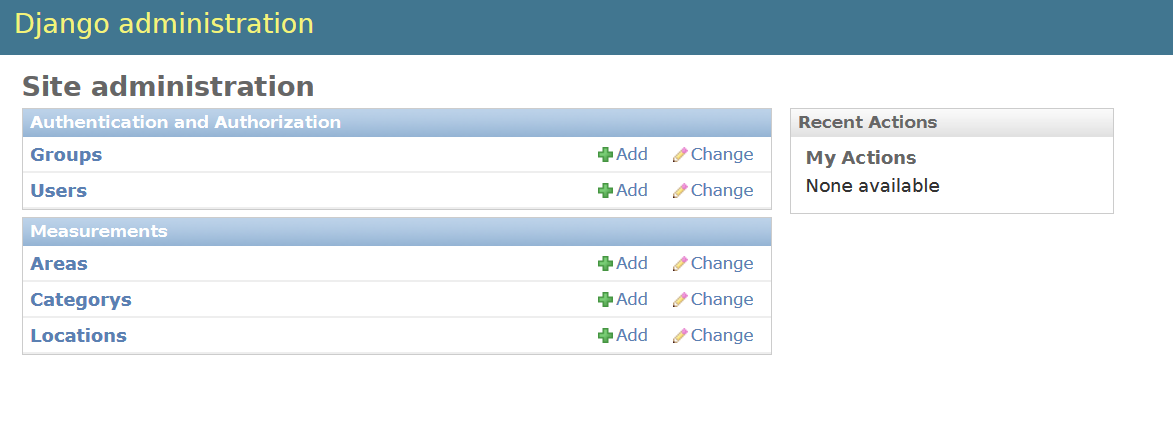
The archive “**Final Project\_management**” contains supplemental code for the project. Unzip the archive so that the management directory is inside the application directory measurements. The files in the this archive add two commands to management.py. These commands are uses to add data to the model database and to clear the data. Give these commands in the management.py console if you are working in PyCharm.

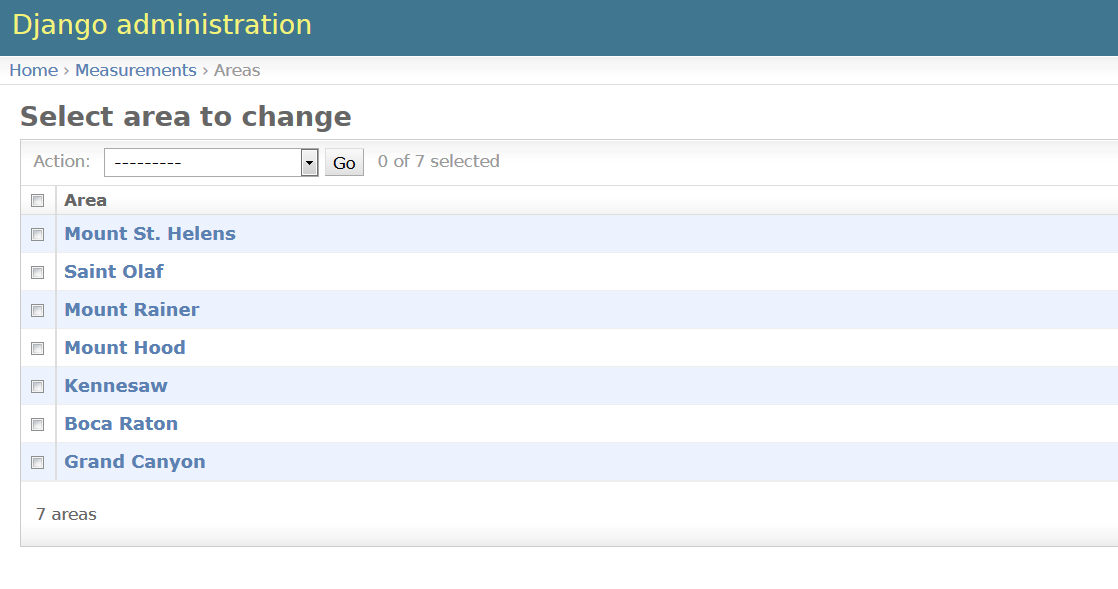
* add\_data: this creates objects and persists them. The data is essentially what you have been using in the measurements examples before.
* clear\_data: this clears all the model instances

For the location of the file: “add\_data.py” and “clear\_data.py”, please refer to the following figure.

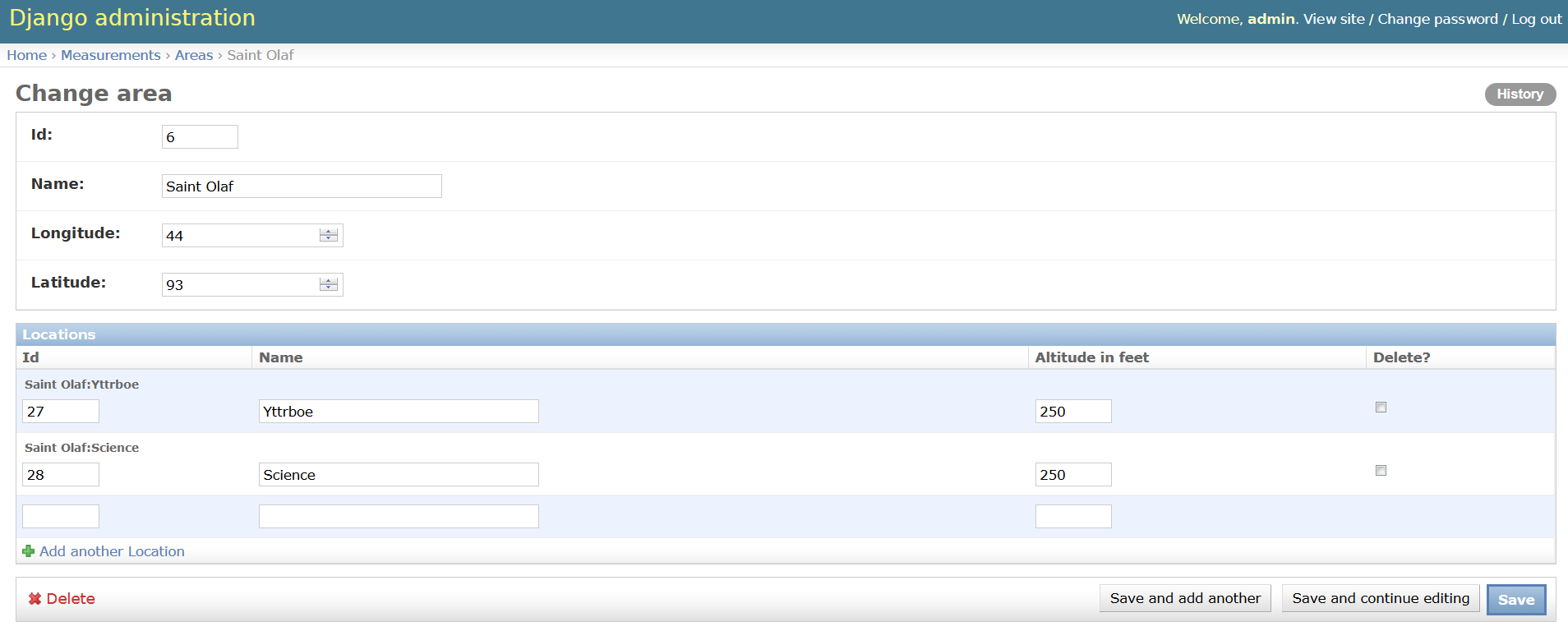


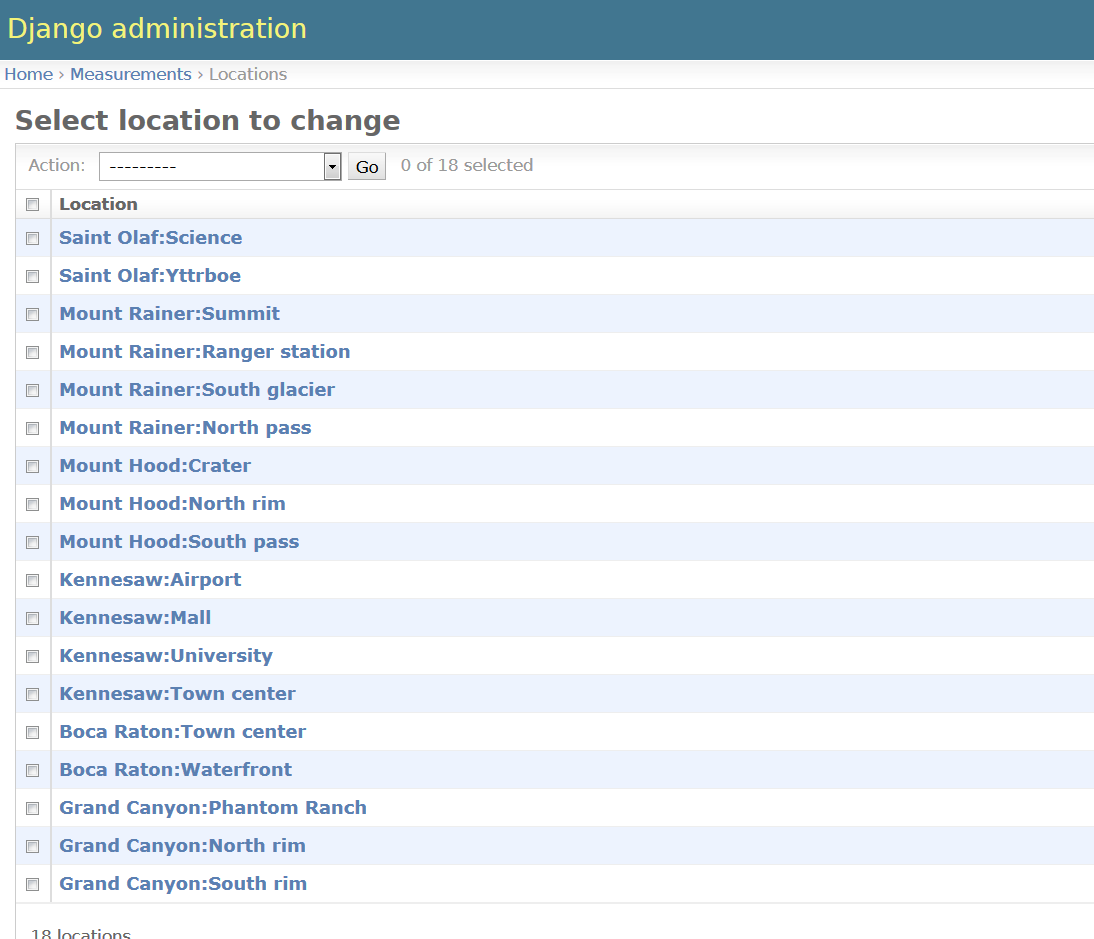
**Administrative Interface (30 points)**

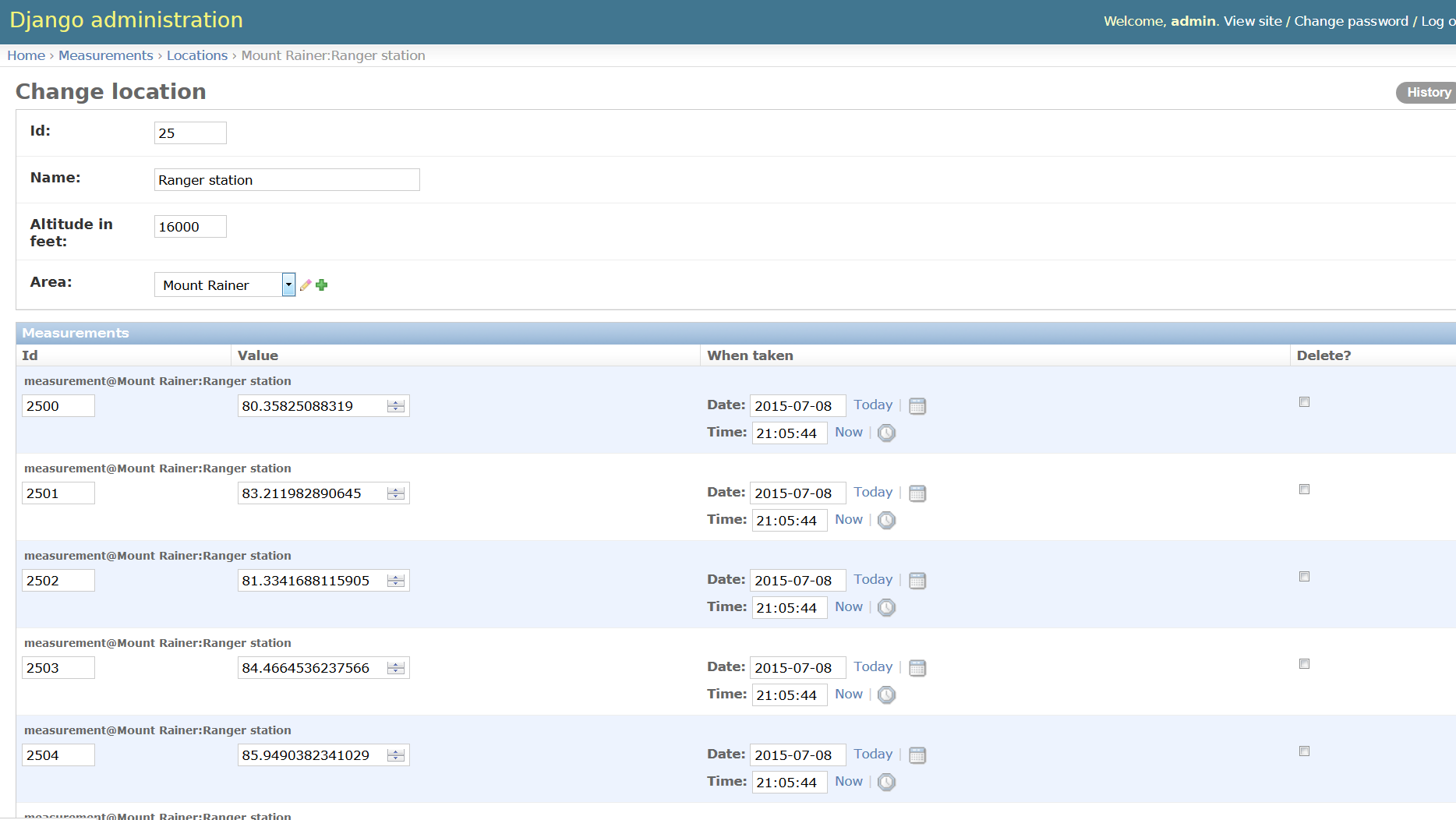
Set up the adiminstrative interface so that Area, Category, and Location are listed on the main admin page.

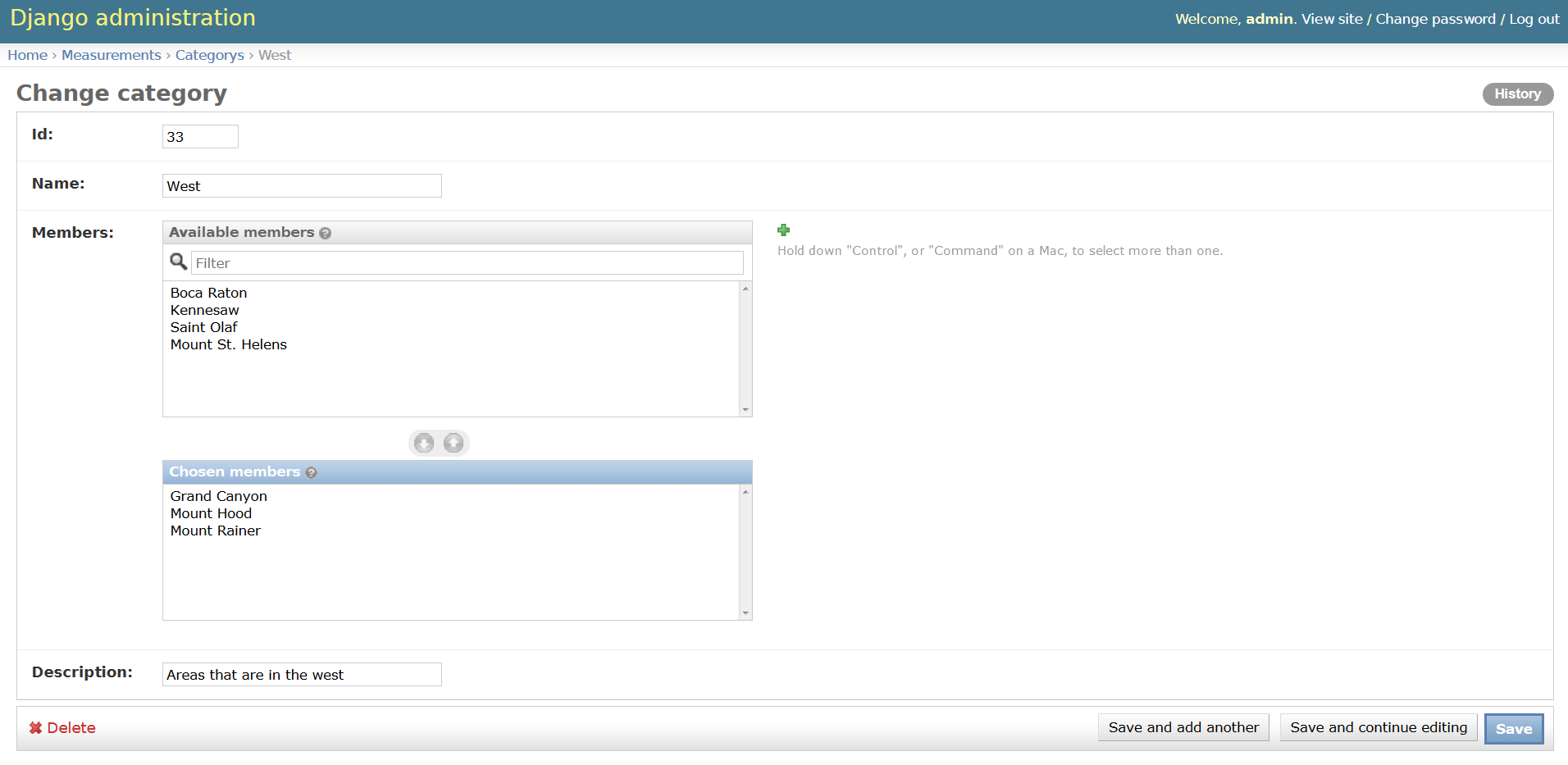
Here is a list of the areas set up by the management commands provided above.

Here is how one area will look. Note that locations are listed here as well as on the main admin page.



Here is the list of locations reached from the main admin page. 

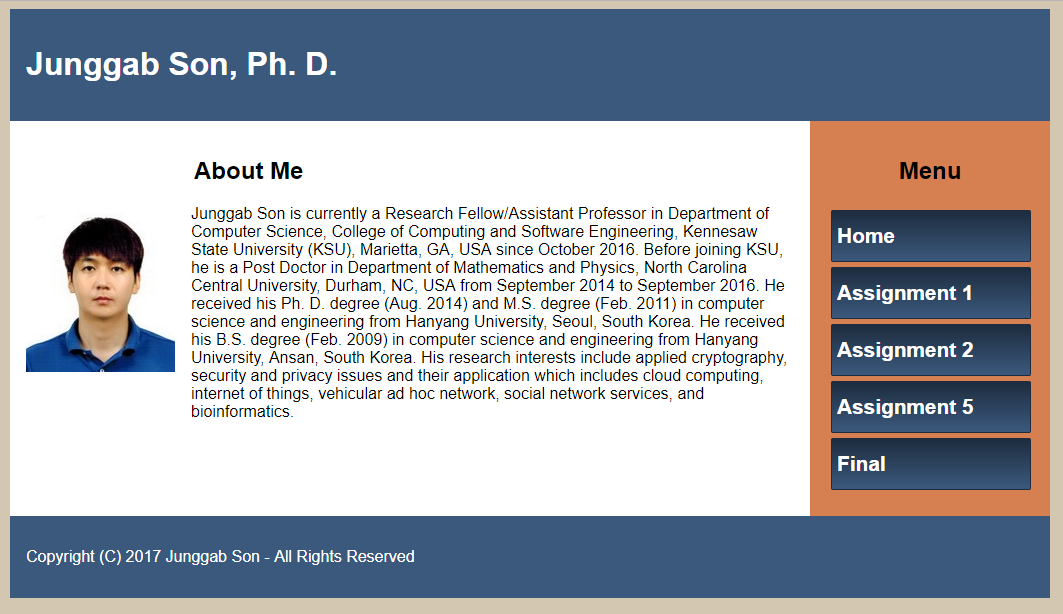
Here is one location with the associated measurements listed below. 

Here is how one category looks. Notice the filter\_vertical option for specifying members. 

**View (35 points)**

**Main page (20)**

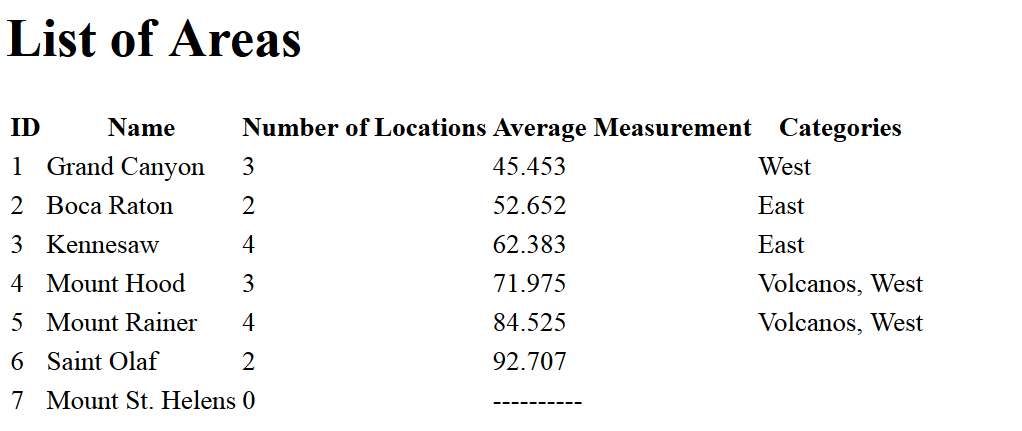
The app supports a single view, at the path / (root). This view presents a main page. For instance:



* All the buttons should properly road web pages, and every page should have a button to go back to the home page (10 points)
* The Assignment 5 page presents a weather information tabular (set a zipcode=30060) (10 points).

**Final page (15)**

This page presents a tabular summary of the data for all areas as shown at below:



Formatting floating point numbers in Django is handled differently from Jinja2. In Django, use the floatformat filter.

**Submission**

Archive the entire project and turn it in.

**Testing**

* Your project will be unarchived.
* The database file will be cleared
* The make\_migration and the migrate commands will be run
* The add\_data command will be run
  + If the management command folder is not present, it will be added (no penalty)
  + It is part of the requirements of this assignment that the add\_data command be supported by your implementation of the model classes.
* The runserver command will be run
* The admin interface will be examined
* The path / will be examined.

Your project will be tested on either Windows or Linux Mint. If you think there could be a problem, please put a comment in the dropbox specifying which OS to use.

If the project contains literal absolute file paths, e.g., c:/Users/Sue/db.sqlite, the project will probably not work properly.  
I will attempt to correct such problems, but substantial credit could be lost depending on the amount of time necessary.