COMPSCI 326
Project 2: Data Model and Views

Overview
The goal of this project is to design and implement a data model for your team web application and implement views that will dynamically render the application views in the browser with mock data from the database.

Goal #1: Your team must implement your data model using the object relational mapping (ORM) tools provided by Django. You must also populate your database with mock data to be used by the views. You can use the Django admin interface to do this or programmatically.

Goal #2: Your team must implement URL mappings and routes to link and display your user interface (UI) views from Project 1. Your UI views must use the mock data in your database and Django templates to dynamically render the user interface to your team's web application.

Please read the Final Project Specification to understand the requirements for the final submission. This will help guide you in the decisions you make for your data model and its implementation, the user interface, and general design goals.

Part 0: Project 2 Startup Details
The following should be performed by exactly one team member. Our suggestion is that your team arrange a meeting time and have one team member run through the steps below.
In a folder of your choice, go through the following steps.

Step 1: You must create a Django project by running the following command:

django-admin startproject TEAMNAME

You must replace **TEAMNAME** with the name of your team (or some shortened version of it).

Step 2: You must create a Django web application inside your Django project by running this command:

python3 manage.py startapp APPNAME

You must replace **APPNAME** with the name of your application.

Step 3: Follow the instructions in Creating a Skeleton Website to setup additional configuration information including registering your application, timezone, etc.

Step 4: After you have created the project and web application skeleton you should run it to make sure it is functioning properly. Once you have verified that it is functioning properly one team member must copy the Django project folder into your team's github repository folder and commit and push your code to Github.
Make sure you check that Github has successfully received your project and then have all team members pull the updated Github repository to everyone's local host. Every team member should then verify that they can run the skeleton repository.

Part 1: Data Model Overview

You must implement a data model using Django's ORM. In particular, you are to follow the same procedure we covered in designing the local library models in the homework to implement your application's data model in Django.

Task #1: Complete a data model diagram for your team's application using the notation discussed in designing the local library models. You must submit a completed diagram of your data model with this submission.

Task #2: Implement the data model using Django's ORM in Python. Your data model implementation must match your data model diagram described in Task #1. You must use the sqlite3 database.

Task #3: Populate your database with mock data using the Django Admin interface or by adding data programmatically. You must add a sufficient amount of data to your database to adequately populate your application's user interface.

Part 1 Details

Step 1: Complete the data model diagram. We recommend using presentation software such as Keynote, Powerpoint, or Google Slides to complete the diagram as you will need this as part of your final presentation at the end of the semester. You must submit a PDF version of your diagram with this project submission.

Step 2: Your team must then implement your data model using Django's Object Relational Mapping (ORM). You should reference Using Models to help guide you in adding Python classes to represent your data model. It will be helpful to add your data model incrementally to make sure you are doing things correctly. This will require coordination with your team.

As you add each new data model class make sure to periodically perform a database migration:

```
$ python3 manage.py makemigrations
$ python3 manage.py migrate
```

Again, it is best to do this incrementally, with small additions, to verify that everything is working.

Step 3: Your team must register your models with Django in order to have them accessible in the Django admin site. This will allow you to easily verify that your models are correct and working, add mock data, and allow course staff to review your model and data in the admin site.

Step 4: Your team must create a superuser account. To do this you must run the command:

```
$ python3 manage.py createsuperuser
```

Your must use the following credentials:

username: compsci326
password: compsci326

Step 5: Use the Django admin site to add mock data. Your team should add enough mock data to allow your views to populate your templates with a sufficient amount data. You are welcome to add list and detail views to your application to customize your admin site views to make it easier to view and add more data. Please consult the Django Admin Site tutorial in the Advanced Configuration section as well as the Generic List and Detail Views tutorial for guidance.

Notes: Make sure you commit and push to Github constantly to ensure that your work is in the repository.

Part 2: Templates and Views Overview

You must implement the URL mappings for your application, view functions/classes, and template files to render your applications user interface (UI). Your views must be dynamically rendered with data from your data model.

Task #1: Implement the URL mappings for your team's application in your web applications urls.py file. Your URL mappings should follow the procedure outlined in the Creating our Home Page from MDN.

Task #2: Convert the mock UIs your team submitted from Project 1 into template files as described in Creating our Home Page from MDN. You must use the Django template language to accomplish this task.

Task #3: Implement the view functions/classes to access your data model and populate the template files for each of your UI views. You should then link your view functions with the URL mappings from Task #1 in order to access them from the browser.

Part 2 Details

Setup: Before you begin implementing your templates and views you need to setup the proper directory for your application. In particular, you will need to create a **templates** directory in your web application directory in your Django project folder. Please see the Template section in the Creating Our Home Page tutorial for guidance.

Step 1: Your team should create a "base" template file from which your user interface templates can "extend". You are welcome to use the format outlined in the Creating Our Home Page tutorial, extend that one, or define your own.

Step 2: The following steps should be performed for each mock user interface (UI) view your team defined in Project 1. If your application views have changed since then then you should note that as part of the writeup. We expect the application to evolve from one project submission to the next.
For each mock user interface:

1. Create a new template file in the proper folder. It will help if you start by copying your HTML from your mock UI submitted for Project 1. You should then replace the mock data parts of your template with *template tags* and *template variables* which will be filled in by a view's rendering process. We suggest as a first pass to only replace some of the mock data to make sure things are working.
2. Implement a new view function/class in your application's **views.py** file for the UIs template file. This view function should request the data from your data model and pass along the data in the context of the render function. We suggest to start simple by only passing in one or two context data to make sure things are working.
3. Implement a URL mapping in the application's **urls.py** file for the UIs view function/class. You should choose a URL path that makes sense for this UI view.

This step should be an iterative process where you slowly add additional details to your UI views. Start by implementing a single URL, View, Template for a single UI page. Start as simple as possible. It will save you lots of time. Note: Do not forget to add an include in the Django project's urls.py file so that Django will properly route HTTP requests intended for your web application. (See Creating a Skeleton Website for more details)

Part 3: Team Write Up

Your team must submit a written 1 page document with the following sections:

- **Overview:** A brief overview of your application. Please highlight any changes from your project proposal since the Project 1 submission.
- **Team Members:** A list of your team members
- **Github Repository:** A link to your team's Github repository
- **Design Overview:** A brief design overview of your data model as implemented in Django, the important URL routes, and the implemented UI views.
- **Problems/Successes:** A brief overview of the problems and successes your team encountered. This includes team communication problems/successes, what worked and what didn't, implementation problems/successes, what worked and what didn't, and what your team can do to improve collaboration and implementation for the next Project submission.

NOTE: This has a 1-page limit. Be brief, but be concise. Use the space wisely.

Part 4: Individual Write Up

Each team member must submit one or two written paragraphs attached to the team writeup that describes exactly what you did as an individual and the percentage of work you contributed to the project as a whole.

By including your write up in the overall team writeup it makes it clear that the team as whole has signed off on your contributions.

It doesn't help you to exaggerate your contributions. If we find that you were not honest in this section it falls under academic dishonesty. So, it is in your best interest to be honest. This is a team-based course which means you are evaluated on your contribution to your team and the semester project.

Notes

Running the Application

The software artifact you submit must be a functioning Django application that will allow the course staff to easily run your web application and visit it through a web browser (we will use Chrome). That is, we should be able to run: **python3 manage.py runserver** and view your app in Chrome. If you are unsure if this will work you should zip up your project and try to run it from a different folder or machine to see if this works.

Not Required

- You should not include any login/logout views as your application currently does not support multiple users - this will come later.
- You should not include any way to update your data model from the browser. That is, you should not have any functioning forms - this will come later.
- You should not include any fancy styling unless you have time to do so.

Required

- Your application should provide basic navigational features. We should be able to navigate to each of your application's views to evaluate your work. If the design of your application requires future features to complete this you should provide a mock navigation menu to allow us to view each of your UI views easily.
- You must include a populated database in your submission.
- You must submit a running application.

Static Assets
If your application requires static assets such as custom CSS or images you must setup a static route in your application to serve such files. You can review how to do this from past homework assignments. In particular, how to setup the static URL mapping (Creating a Skeleton Website) and how to reference static files in a template (Creating our Homepage).

Mocking versus Feature Complete
We recognize that your team may have bigger ideas. Remember, this is not a final submission. The goal with this project is to create a prototype of what your application might look like and to incorporate URL mappings, views, and templates to demonstrate the capabilities of your data model and linking between views. Keep it simple, make it work, there is time later to be fancy and add additional functionality.

Rubric

Part 1 Data Model (35 Points)
- 10 points for a completed data model diagram.
- 10 points for data model entities implemented in Python/Django.
- 5 points for a functioning admin site.
- 10 points for the addition of mock data.

Part 2 Templates and Views (35 Points)
- 10 points for the implementation of template files.
- 15 points for the implementation of view functions/classes.
- 10 points for the implementation of URL mappings.

Part 3 Team Write Up (15 Points)
- 15 points for the team writeup.

Part 4 Individual Write Up (15 Points)
- 15 points for the individual writeup.

Submission
Your submission will include only a link to your team's Github repository. In your repository you must have:

- A folder for your Django application which you created during the project 2 startup.

- A folder called **writeups** which contains a PDF named "project-02-writeup.pdf". The PDF must contain the team writeup and the individual writeup for each team member in a single document.

With the proper repository structure in place you will need to "tag" this version of the repository so the course staff can evaluate your team project at this point. To do this, one team member must perform the following git commands:

```
git tag -a project2 -m "project 2 submission"
git push origin project2
```

This will "tag" your git repository and push it to Github to allow us to review your submission. This will also allow your team to continue work on your project without interfering with our grading process.