

# SMARTBANK Database

Christopher Tao, Emmie Kao, Dr. Ryan McGorty,  
Dr. Rae Anderson

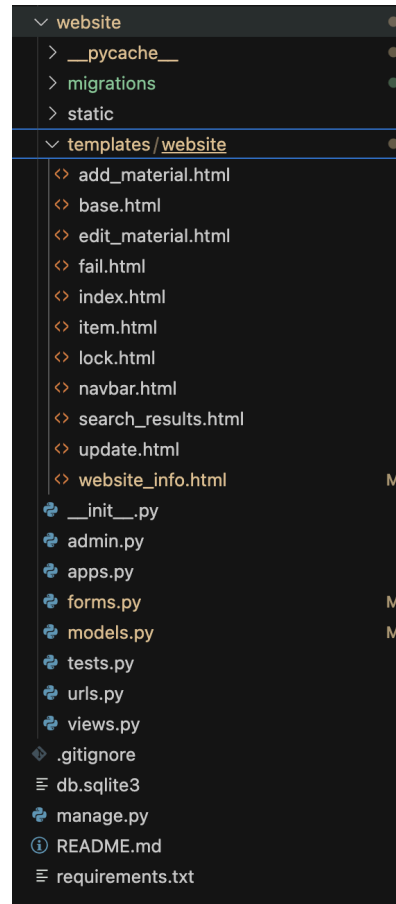
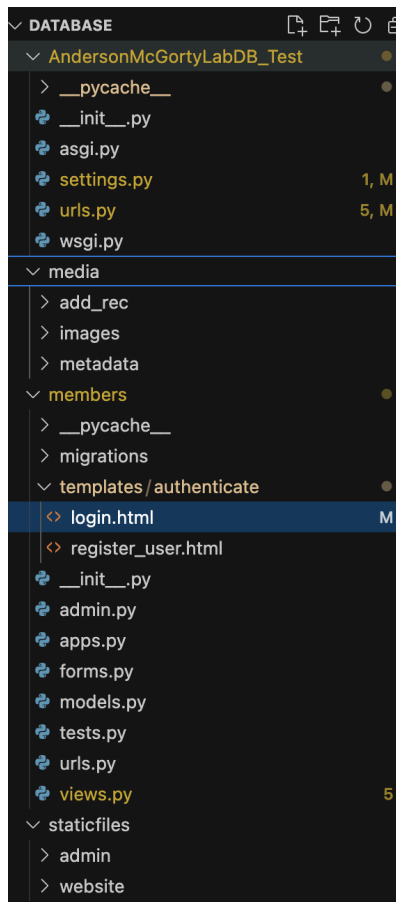
07/24/24

# 1. Introduction

The SMARTBANK database aims to solve the issue of cluttered and disorganized data. As a minimum viable product (MVP), the SMARTBANK website is expected to be able to store videos and descriptions and display them when requested. It will consist of two main pages, one that displays a table of the data in SMARTBANK, and another that provides more information on the experimental protocol and procedures behind each database entry. In this document, we will explain the framework behind the program as a MVP as well as future developments.

## 2. File Structure

Below is an overview of the file hierarchy. Because there are multiple files with the same name, each time those files are referenced the parent folder will also be listed.



## 3. File Overview

There are several important files and folders that require descriptions.

### 3.1 Models.py

Models.py holds a copy of all of the entries inside a database. It allows for smooth manipulation of the database without resorting to opening a POSTGRESQL interface in the terminal. After modifying models.py, the changes can be migrated to the database by typing in the python command “python3 manage.py makemigrations”, which will create a new migration. This modification can be finalized by adding in the line “python3 manage.py migrate”.

While there are two models.py files, this document will mainly cover the website/models.py.

### 3.2 Forms.py

The programmer can customize the fields that are displayed in interactive forms, as well as the input method. Registration forms can be found inside members/forms.py, while database edits can be modified in website/forms.py.

### 3.3 Views.py

All of the functions and logic calls in the website are located inside views.py. Creating accounts and logging in require navigating through members/views.py, while adding in, modifying, or showing data points use website/views.py.

### 3.4 Urls.py

While views.py contains all of the functions, urls.py makes the functions usable. Urls.py connects each function in views.py to a name and endpoint. Whenever a name is called, such as “search” or “add\_material”, the respective function is called. Other than using the name, the endpoints can also call a function. Adding a “search/” or a “add\_material/” at the end of smartbankdb.org will result in the same function calls as above.

There are three url pages: members/urls.py, website/urls.py, and AndersonMcGortyLabDB\_Test/urls.py, which will be referenced as urls.py. When endpoints are called, the program will search urls.py for the endpoint. The other two url files are meant to add readability to the program, and these files are called by urls.py.

### 3.5 Templates

Each template folder holds the html files that make up the frontend of the program. Most of the files are located inside website/templates, although the registration and login pages are hosted inside members/templates.

## 4. Software Architecture

### 4.1 Backend

The backend handles any function calls and logic behind the frontend: it does what the user is not able to see. This includes getting or changing database entries and searching by a specific criteria.

For this website, the backend is separated into two main folders: websites and members. While the website folder contains the necessary files for database management, the member's folder provides access control to the database.

#### 4.11 Website Backend Development

The website takes advantage of the functionality of django to create a simple interface. The backend uses django's model structure, specifically the table named Softmatterdata, to modify and update the database structure. The model structure is combined with a forms page that specifies the input methods for each element. The forms are styled through the widgets section and filled out when functions are called.

Website/views.py contains all of the functionality behind the buttons and forms. Written in python, it is a collection of functions that edit or display certain pieces of information. Notably, the website calls the "index" function when displaying the home page, and it calls "materialInfo" when creating the additional-information page. These functions are called through a list of endpoints inside urls.py. The function calls and usability will be explored further in the front end section.

```
class Softmatterdata(models.Model):
    identifier = models.AutoField(primary_key=True)
    composition = models.CharField(max_length=255)
    method = models.CharField(max_length=255)
    name = models.ForeignKey(User, blank=True, null=True, on_delete=models.SET_NULL)
    acquired = models.DateField()
    lastupdate = models.DateField(auto_now=True)
    doi = models.CharField(max_length=255, blank=True, null=True)
    summary = models.TextField(blank=True, null=True)
    lock = models.BooleanField(blank=True, null=True, default=False)
    sample_image = models.ImageField(null=True, blank=True, upload_to="images/")
    meta_data = models.FileField(null=True, blank=True, upload_to="metadata/")
    additional_resources = models.FileField(null=True, blank=True, upload_to="add_rec/")

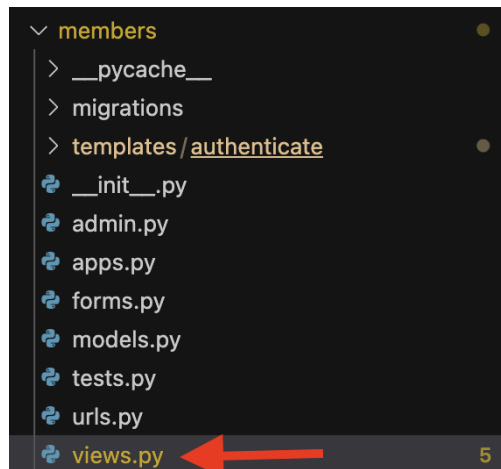
    class Meta:
        managed = True
        db_table = 'softmatterdata'
```

#### 4.12 Login and Registration

The members folder mimics the structure of the website folder with a model views structure. Rather than creating personal models for the users, the program takes advantage of Django's built-in user model, which includes labels such as username, password, email, and

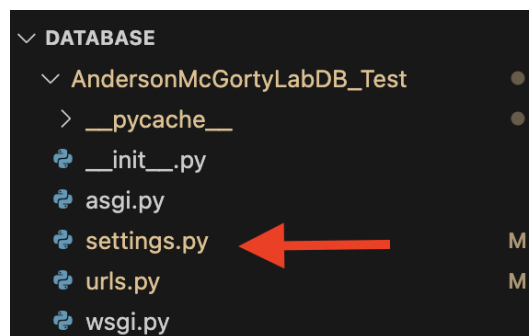
more. The forms file filters through all of the criteria in the users class and shows the mandatory fields, which are chosen by the developer.

This form is called through members/views file. This views file contains two functions: login and registration. The registration file calls the form for the user to fill out, while the login feature finds a corresponding user given the username and password. Because the website uses Django's built-in model for a user, the password has very specific regulations, including a minimum length and a minimum number of unique characters. Given that the inputted password does not meet all of the requirements, the user will be met with a popup message. These functions are called through Django endpoints, similar to the website function calls.



#### 4.13 Database Connectivity

The connection to the database, as well as any other stored information, is set up in the "settings.py" file. The Database section of the settings holds the keys and passwords necessary to access the database. The bottom of the file contains several variables labeled "URL" or "ROOT". These paths point to important folders that store images and file content.



#### 4.14 Image and File Storage

The website model contains several variables that are either called "ImageField" or "FileField". However, it is unfeasible to directly store these variables into the database. Images and Gifs are especially difficult due to their large file size and POSTGRESQL's lack of image support. Therefore, rather than saving the raw files into the database, the database stores a url that points to the original file. These files are located inside the media folder, and the URL and

ROOT variables in the settings allow the program to dictate exactly which folder, and to which url, each file will be saved.

```
sample_image = models.ImageField(null=True, blank=True, upload_to="images/")
meta_data = models.FileField(null=True, blank=True, upload_to="metadata/")
additional_resources = models.FileField(null=True, blank=True, upload_to="add_rec/")
```

## 4.2 Website Frontend Development

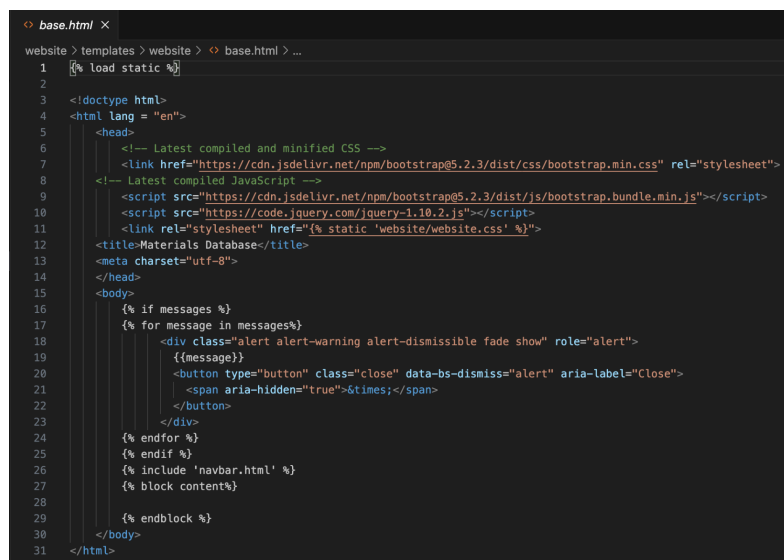
The frontend consists of all of the pages that a user can see and interact with. The frontend comprises of html files, which are located inside the folders labeled “templates.” Each file represents a different page on the website.

### 4.2.1 MVP frontend

The templates that are essential to the website are located inside the website folder. There are several important files: navbar.html, base.html, and index.html. The navbar.html creates the bar at the top of the page. The navbar lets the user navigate to different pages when they click the buttons.

The base.html serves as the support that the rest of the front end is built off of. It imports all of the necessary components and creates a base template that all of the other html files are built off of. All other html files use the line `{%extends 'base.html'%}` to reference the base template.

The index html file also extends the base file. It is the homepage that users will see when they first enter the website. It displays the materials inside the website and provides links to pages with more information.

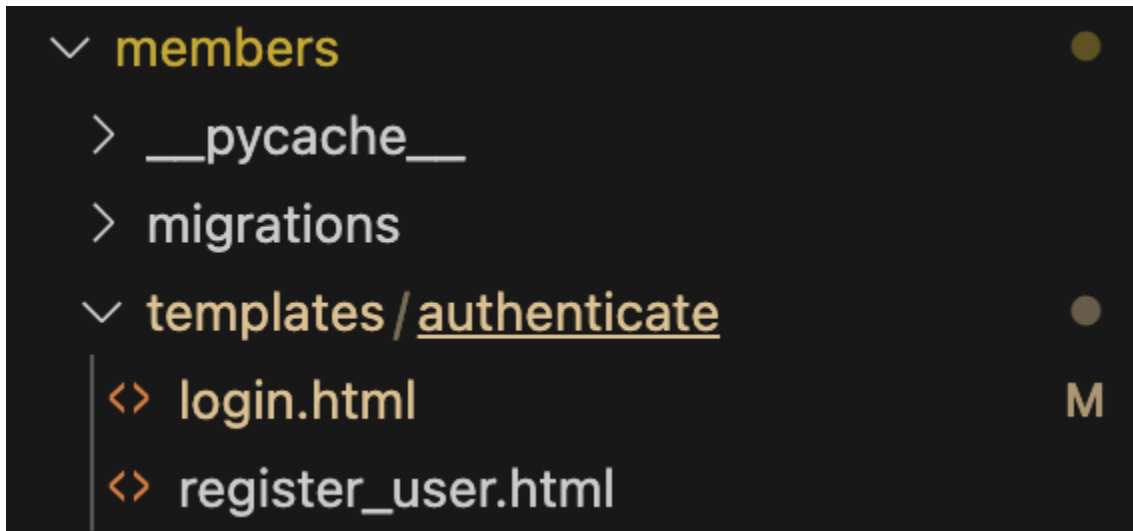


```
<> base.html X
website > templates > website > <> base.html > ...
1  {% load static %}
2
3  <!doctype html>
4  <html lang = "en">
5
6      <head>
7          <!-- Latest compiled and minified CSS -->
8          <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.3/dist/css/bootstrap.min.css" rel="stylesheet">
9          <!-- Latest compiled JavaScript -->
10         <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.2.3/dist/js/bootstrap.bundle.min.js"></script>
11         <script src="https://code.jquery.com/jquery-1.10.2.js"></script>
12         <link rel="stylesheet" href="{% static 'website/website.css' %}">
13         <title>Materials Database</title>
14         <meta charset="utf-8">
15     </head>
16     <body>
17         <div class="alert alert-warning alert-dismissible fade show" role="alert">
18             <div class="alert alert-warning alert-dismissible fade show" role="alert">
19                 <span style="float:right">
20                     <button type="button" class="close" data-bs-dismiss="alert" aria-label="Close">
21                         <span aria-hidden="true">&times;</span></button>
22                 </div>
23             </div>
24         </div>
25     </body>
26     <include 'navbar.html' %>
27     <block content%>
28
29     </block %>
30 </body>
31 </html>
```

### 4.2.2 Login and Registration

The html files for logging in and registration can be found inside the members folder. The registration file uses the class in members/forms.py to display input options. However, the login html uses a different approach. Rather than directly filling out a form, the user fills out two

separate inputs, which will be sent back to the views.py to verify that the inputs are valid. For erroneous values in both registration and logging in, the frontend creates a popup message that tells the user that there was an error with their responses.



#### 4.23 Styling

The frontend uses bootstrap as well as a custom css file to modify the display of the website. More information about bootstrap can be found here: <https://getbootstrap.com/>.

#### 4.3 Database

DigitalOcean hosts the database that holds all of the information. The database uses a POSTGRESQL framework and its credentials can be found in settings.py and environmental variables can be modified in DigitalOcean.

A database contains multiple tables which can be edited through models.py.

**App-Level Environment Variables**

All components will have access to these variables at build time and runtime. If a component has a variable with the same key, the component's value will override the app-level value. [Learn More](#)

[Bulk Editor](#)

Keys	Values			
DISABLE_COLLECTSTATIC	1		<input type="checkbox"/> Encrypt	
DEBUG	0		<input type="checkbox"/> Encrypt	
DJANGO_ALLOWED_HOST	sea-turtle-app- xxxxx2-digitalocean.com		<input type="checkbox"/> Encrypt	
DJANGO_SUPERUSER_EMAIL	softmatterdb@gmail.com		<input type="checkbox"/> Encrypt	
DJANGO_SUPERUSER_NAME	softmatterdb		<input type="checkbox"/> Encrypt	
DJANGO_SECRET_KEY	Encrypted			
DJANGO_SUPERUSER_PASSWORD	Encrypted			

## 4.4 Domains

DigitalOcean offers a website domain and deployment. However, University of San Diego has acquired a dedicated DNS domain, smartbankdb.org, which is created through Bluehost.

Connecting this domain to DigitalOcean required modifying the nameservers on Bluehost to point towards DigitalOcean's DNS servers.

### Advanced Tools ^

Advanced tools offer more control and allow you to customize and manage nameserver settings as well as DNS records such as A, CNAME, and MX records. Restore original nameserver settings to benefit from the suite of Bluehost.com services such as email, websites, and more. [Learn more about DNS Records](#)

<b>Nameservers (DNS)</b> ⓘ <b>MANAGE</b>	<b>Advanced DNS Records</b> ⓘ <b>MANAGE</b> ⓘ
NS1.DIGITALOCEAN.COM	Edits on <b>A(3)</b>
NS2.DIGITALOCEAN.COM	
NS3.DIGITALOCEAN.COM	

Looking for custom nameservers?  
Create them on the [Custom Nameservers](#) section.

## 5. Post Production Features

Although the MVP, along with the registration feature, has been implemented, there are still several enhancements that need to be made. The first priority is email verification and password recovery. Afterwards, modifications to the user interface and security should be implemented to improve the user experience.