

Web programming 1



Contents

1	Introduction	1
1.0.1	server side programming	2
2	Django Intro	6
2.0.1	How does Django Work?	7
2.0.2	Components of Django	9
3	Install Django	12
3.1	prerequisite	12
3.1.1	Python	12
3.1.2	PIP	13
3.1.3	create a project directory	14
3.1.4	Virtual Environment	14
3.1.5	Install venv	15
3.1.6	create venv project	15
3.2	Install Django	16
3.2.1	Check Django Version	17
3.3	First project	17
3.4	test if our project is working	19
3.5	APPS	21
3.5.1	Views	29
4	Add database	35
4.1	django postgresql setup	36
4.1.1	Creating the PostgreSQL Database and Users	37
4.1.2	Install database bindings	39
4.2	Completing database Setup	40
5	Templates	44
5.1	How do templates work in django?	44
5.2	using templates for dynamic websites	46
5.3	Diff Between {{ }} and {% %}	48
5.3.1	{{ Variables }}	48
5.3.2	{% Tags %}	50
5.3.3	{{ value filter }}	52
6	Setup IDE	53
6.1	install vscode	53
6.1.1	install extensions for vscode	54
6.2	setup our project	54
7	college management website	57
7.1	main landing page	57
7.2	static	64
7.3	split up page for better control	65
7.3.1	base html file	66
8	Authentication	72
8.1	What is user authentication?	72
8.2	Web Authentication Methods Compared	74
8.2.1	Authentication vs Authorization	74
8.2.2	HTTP Basic Authentication	75
8.2.3	HTTP Digest Authentication	77
8.2.4	Session-based Auth	81
8.2.5	Token-Based Authentication	85
8.2.6	One Time Passwords	90
8.2.7	OAuth and OpenID	94

8.3	Implement authentication in Django	98
8.3.1	Django website custom user	98
8.3.2	create user model	99
8.3.3	Enable custom user model authentication	101
8.3.4	register user model	101
8.4	create custom admin form	103
8.4.1	register user model	103
8.4.2	make migrations	103
8.5	log back into admin site	109
8.6	Add users to the app	110
8.7	create an app for each type	113
8.8	mainsite web page	113
8.8.1	base html	114
8.8.2	home html	115
8.8.3	include navbar html	116
8.8.4	include footer html	117
8.8.5	include messaging html	119
8.8.6	mainpage urls.py	119
8.8.7	mainpage views.py	120
8.9	user templates and code	120
8.9.1	user forms.py	120
8.9.2	user admin.py	121
8.9.3	user models.py	121
8.9.4	user urls.py	122
8.9.5	project urls.py	122
8.9.6	user views.py	123
8.9.7	user templates-user-login.html	126
8.9.8	user templates-user-logout.html	127
8.10	faculty code	127
8.10.1	faculty view.py	127
8.10.2	faculty faculty-templates-faculty-base.html	128
8.10.3	faculty faculty-templates-faculty-home.html	129
8.10.4	project urls.py	129
8.10.5	project settings.py	129
9	GIT/Versioning	132
9.1	What is git?	133
9.2	git servers and providers	135
9.3	install git	136
9.3.1	MINGW64	136
9.4	Linux git install	137
9.5	initial local git setup	138
9.5.1	windows git bash setup	138
9.5.2	linux setup	140
9.6	getting help with git commands	140
9.7	start git with our project	142
9.8	github	167
9.8.1	create a repository	168
9.8.2	verify or create ssh keys	171
9.8.3	Finish repo upload	175
9.8.4	Django requirements file	176
10	Models	179
10.1	what is model ?	179
10.2	field types	180
10.3	ORM	180
10.3.1	different types of ORM	182
10.4	migrations	183

11 querysets	192
11.0.1 Managers	192
11.0.2 backends	194
11.1 QuerySet API	194
11.1.1 Methods that return new QuerySets	194
11.2 ORM for all teachers and classes	214
11.3 passing data to a queryset on URL	218
12 user data database	219
12.1 database model for college	224
12.1.1 model the database	225
12.1.2 Database design	228
12.2 College registrar page	230
12.2.1 Registrar base.html	232
12.2.2 X-frame-options	241
12.3 revisit our login page	242
12.3.1 Authentication vs Authorization	244
12.3.2 Make changes to user login view.py	246
12.3.3 Use the role to restrict users from accessing paged that their role does not permit .	246
12.4 create a page to add a user	247
12.4.1 add user creation form to our urls.py	249
12.5 crispy forms	254
12.5.1 get all the fields in our form	260
12.5.2 create a standalone script to run parts of django	261
12.5.3 better,faster and improved adduserform	263
12.5.4 error checking and filling in other information	264
12.5.5 Django User Authentication in form	266
12.6 using django debug toolbar	270
12.6.1 remove the navbar javascript from base.html	270
12.6.2 redesign the registrar home page navbar	271
12.6.3 make small changes to our view to disable the adduser link when in adduser	272
12.7 display all users	274
12.7.1 create an allusers view	274
12.7.2 allusers.html	275
12.7.3 all allusers to navbar	276
12.7.4 add urls to urls.py	276
12.8 ORM queryset	277
12.8.1 Allusers Java script and JSON data	280
12.8.2 modify views.py for JSON views	282
12.8.3 new URLs	283
12.8.4 modify query set to sort items	284
12.9 Create the rest of our models	285
12.9.1 optionally register the model with our admin site	288
12.9.2 verify table creation	290
12.9.3 modify model	290
12.10 manually(bulk) populate our classroom tables	295
12.10.1 script to populate database table	296
12.11 create all the other tables and junction tables(ManyToMany)	297
12.11.1 Classes	297
12.11.2 Current ERD	299
12.11.3 teacher model	302
13 upload data	304
13.1 create loading pages for rest of tables	308
13.1.1 users	309
13.2 teacher csv populate	313
13.3 retrieve classes taught from website Using ORM	319
13.4 faculty loading	323
13.5 student registration	328

13.5.1 drop classes	334
14 class-based-views	342
14.1 When to use CBV?	347
14.2 basic templateview UML	348
14.2.1 templateView	348
14.2.2 Basic list view	354
14.2.3 pagination on listview	357
14.2.4 improved pagination	362
14.3 all build in class based views	367
15 gunicorn and nginx	368
15.1 install gunicorn	368
15.2 install httpie for testing	368
15.3 start django as runserver	369
15.3.1 test that django runserver is listening	369
15.3.2 use httpie to test connection	370
15.3.3 start gunicorn	370
15.3.4 create a config file for gunicorn	372
15.3.5 systemd start	374
15.4 nginx gunicorn	377
16 debug tool bar	381
16.1 Installation	381
16.1.1 pre requisites	381
16.1.2 Install the App	382
16.1.3 Add the URLs	383
16.1.4 Add the Middleware	384
16.1.5 Configure Internal IPs	384
17 django Extensions	386
17.1 Install django extensions	386
17.1.1 configuration	386
17.2 generate ERD diagrams	387
17.2.1 Install pre req	387

preface

Many images and pictures were obtained from: <https://pixabay.com/>
under the free for commercial use, no attribution required.



Introduction

Server-side programming refers to code that runs on the server, as opposed to client-side programming which runs on the user's computer. Some key aspects of server-side programming:

- It involves writing code to handle server logic like request handling, data access, authentication, etc. Popular server-side languages include PHP, Python, Ruby, Java, C# etc.
- It can interact with databases like MySQL, MongoDB etc. to store, retrieve or update application data.
- It allows serving dynamic content to users, like data from databases, generated on-the-fly rather than static HTML pages.
- Some common web application frameworks used for server-side development are Express (Node.js), Laravel (PHP), Django (Python), Ruby on Rails etc.
- The code runs in response to requests from client-side pages/apps. For e.g. loading a page triggers server-side code to fetch required data and generate the page.
- Server-side code is executed on the server before the page is sent to the client. The client only receives the resulting outputs to display.

In summary, server-side programming generates the logic and dynamic content of a website or app on the backend, which interacts with the databases and delivers the results to the client-side for display. It is a core component of web development alongside client-side programming.

1.0.1 server side programming

Web browsers(web clients) communicate with web servers using the HyperText Transfer Protocol (HTTP/S). When you click a web link on a web page, submit a form, or run a search, an HTTP/S request is sent from your browser to the target server.

The request includes a URL identifying the affected resource. This mechanism defines the required action (for example, to get, delete, or post the resource). It may include additional information encoded in URL parameters (the field-value pairs sent via a query string), as POST data (data transmitted by the HTTP POST method), or in associated cookies.

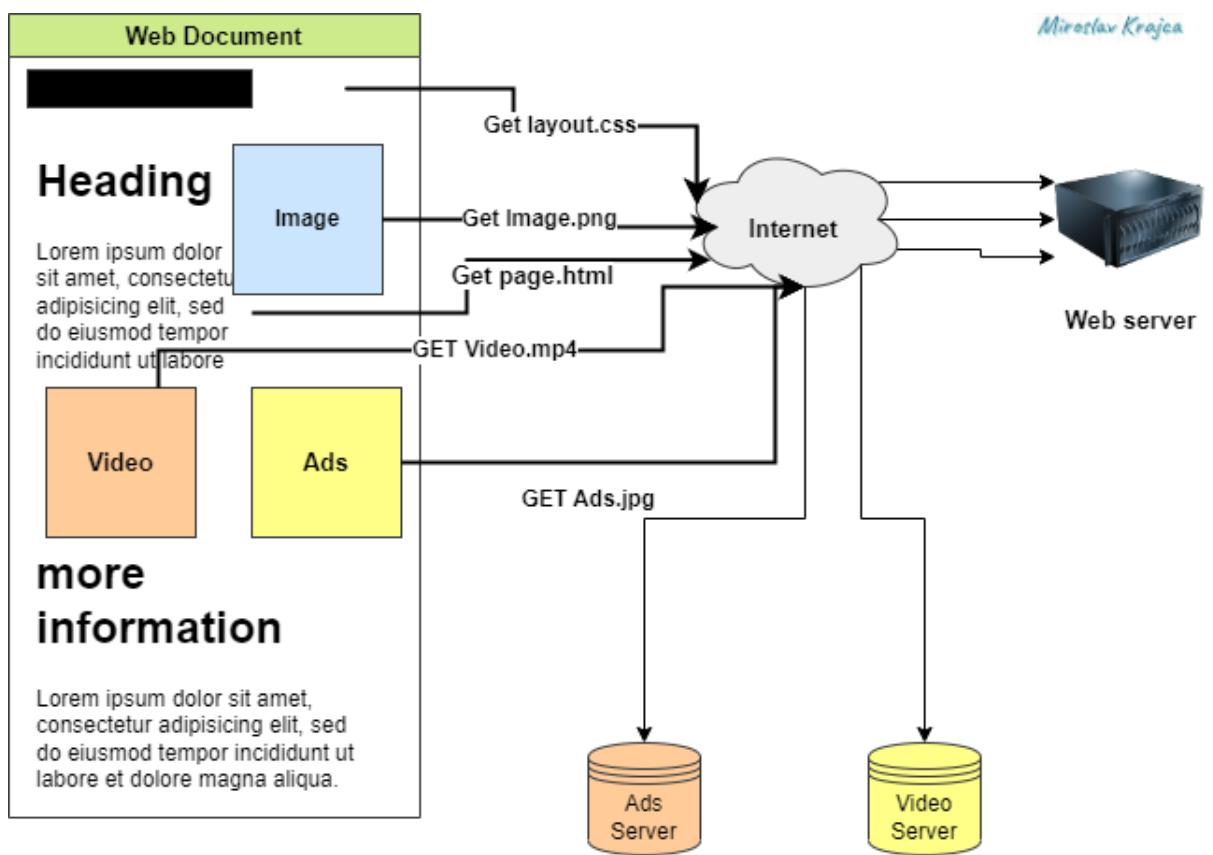
- **GET:** A get request is one of the most common web requests that can be made. This type of request is made on behalf of a client when it's seeking data from a specified resource. Once the client has requested a resource, the server will process the request, get the information or data, and send it back to the client.
- **POST:** This is another common type of web request, but the purpose of POST seeks to create or update the subordinate requested resources. This request does not store additional data but, instead, per request, must be updated and refreshed or resubmitted. Furthermore, each time this request is made, it is clear to the user that it must be resubmitted.
- **PUT:** This method allows the target URL to be entirely replaced with a new resource. PUT should be utilized to replace or overwrite a resource that the client is aware of.
- **DELETE:** This method sends a request to the server to delete a resource. While this is a possibility, it is not the most preferred choice.

- **PATCH:** This method is simply used to modify a specific part of the resource. While it is similar to the PUT method, PATCH aims only to update or modify rather than replace.

Web servers wait for client request messages, process them when they arrive, and reply to the web browser with an HTTP response message. The response contains a status line indicating whether or not the request succeeded (e.g., "HTTP/1.1 200 OK" for success).

The body of a successful response to a request would contain the requested resource (e.g., a new HTML page or an image), which the web browser could then display.

Example of a web client getting information for a web page to be rendered.

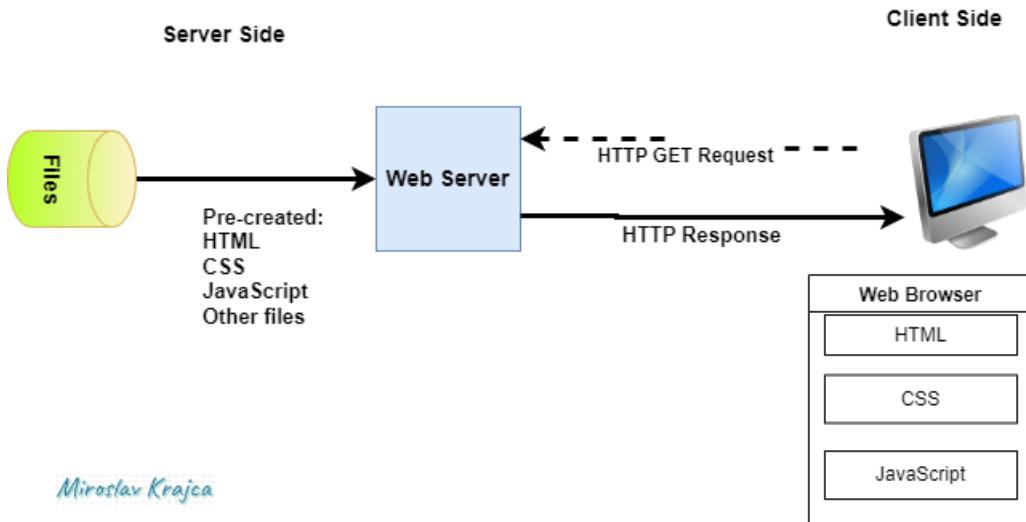


1.0.1.0 static sites

The diagram below shows a basic web server architecture for a static site (a static site is one that returns the same hard-coded content from the server whenever a particular resource is requested). When a user wants to navigate to a page, the

browser sends an HTTP "GET" request specifying its URL.

The server retrieves the requested document from its file system and returns an HTTP response containing the document and a success status (usually 200 OK). If the file cannot be retrieved for some reason, an error status is returned (see client error responses and server error responses).



1.0.1.0 Dynamic sites

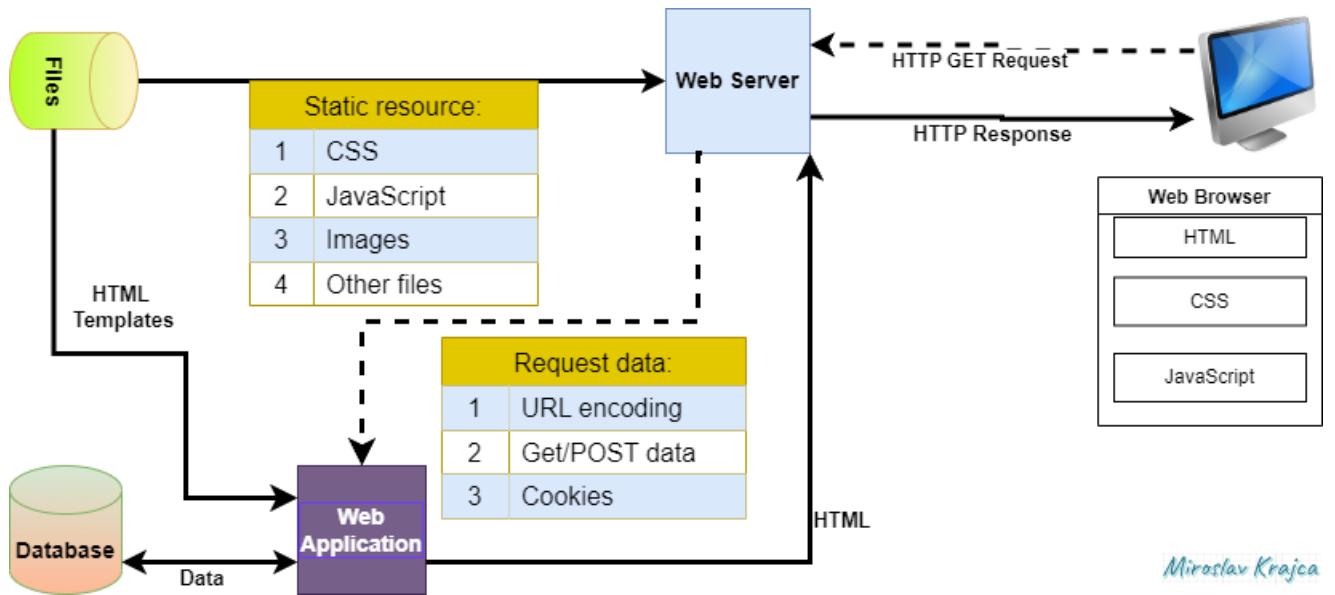
A dynamic website is one where some of the response content is generated dynamically, only when needed. On a dynamic website, HTML pages are typically created by inserting data from a database into placeholders in HTML templates (this is a much more efficient way of storing large amounts of content than static websites).

A dynamic site can return different data for a URL based on information provided by the user or stored preferences. It can perform other operations to return a response (e.g., sending notifications).

Most of the code to support a dynamic website must run on the server. Creating this code is known as "server-side programming" (or sometimes "back-end scripting").

The diagram below shows a simple architecture for a dynamic website. As in the previous diagram, browsers send HTTP requests to the server, and then the server processes the requests and returns appropriate HTTP responses.

Requests for static resources are handled similarly to static sites (static resources are any files that don't change — typically: CSS, JavaScript, Images, pre-created PDF files, etc.).



Requests for dynamic resources are instead forwarded to server-side code (shown in the diagram as a Web Application). For "dynamic requests" the server interprets the request, reads required information from the database , combines the retrieved data with HTML templates , and sends back a response containing the generated HTML.



django

Django introduction

<https://www.djangoproject.com/start/overview/>

What is Django?

Django is a high-level Python web framework that encourages rapid development and clean, pragmatic design. Built by experienced developers, it takes care of much of the hassle of web development, so you can focus on writing your app without needing to reinvent the wheel. It's free and open source.

Django's primary goal is to ease the creation of complex, database-driven websites. The framework emphasizes reusability and "pluggability" of components, less code, low coupling, rapid development, and the "don't repeat yourself" principle. Python is used throughout, even for settings files and data models.

Here are some of the features that Django provides:

- A robust Object-Relational Mapping (ORM) layer to interact with your database, like MySQL, PostgreSQL, Oracle, etc.
- The automatic admin interface, a powerful automatic GUI interface for your models, is dynamic and requires little to no configuration for basic use cases.
- URL routing.

- Template engine, which is a way of generating HTML (or other markup) using Python code.
- Form handling.
- Authentication support.
- Caching, Middleware, Serialization, and more.

Django also includes a lightweight web server for development use. However, industry-standard web servers like Nginx or Apache are commonly used for deploying Django in a production setting.

Django was invented by Lawrence Journal-World in 2003, to meet the short deadlines in the newspaper and at the same time meeting the demands of experienced web developers.

Django is especially helpful for database-driven websites.

2.0.1 How does Django Work?

Django uses a design pattern similar to the Model-View-Controller (MVC) pattern you might see in other web frameworks. However, the Django framework calls this the Model-View-Template (MVT) pattern because of the way it deals with the controller part. Here's how it works:

- **Model:** The model is going to act as the interface of your data. It is responsible for maintaining data. It is the logical data structure behind the entire application and is represented by a database (generally relational databases such as MySql, Postgres). The model defines the structure of the database, the field types, possibly default values, validation behavior, and more. In Django, the data is delivered as an **Object Relational Mapping (ORM)**, which is a technique designed to make it easier to work with databases.

The most common way to extract data from a database is SQL. One

problem with SQL is that you have to have a pretty good understanding of the database structure to be able to work with it.

Django, with ORM, makes it easier to communicate with the database, without having to write complex SQL statements.

The models are usually located in a file called **models.py**.

- **View:** The view is where you put the logic of your application. It will access the model and tuck the data into a template. Views are a bridge between models and templates. In Django, views process the request and return a web response. The web response can be the HTML contents of a webpage, a redirect, a 404 error, an XML document, an image, or anything else you can think of. The views are usually located in a file called **views.py**
- **Template:** The template is a HTML file mixed with Django Template Language (DTL). The presentation layer is taken care of by Templates. This is where the user sees and interacts with. A template is rendered with a context. Rendering substitutes variable values into the template to generate a string. This string can be a HTML page or any other type of document. The templates of an application is located in a folder named **templates**

```
<h1>OCCC Main page</h1>
<p>Course name is {{ coursedescription }}.</p>
```

- **URLS:** Django also provides a way to navigate around the different pages in a website.

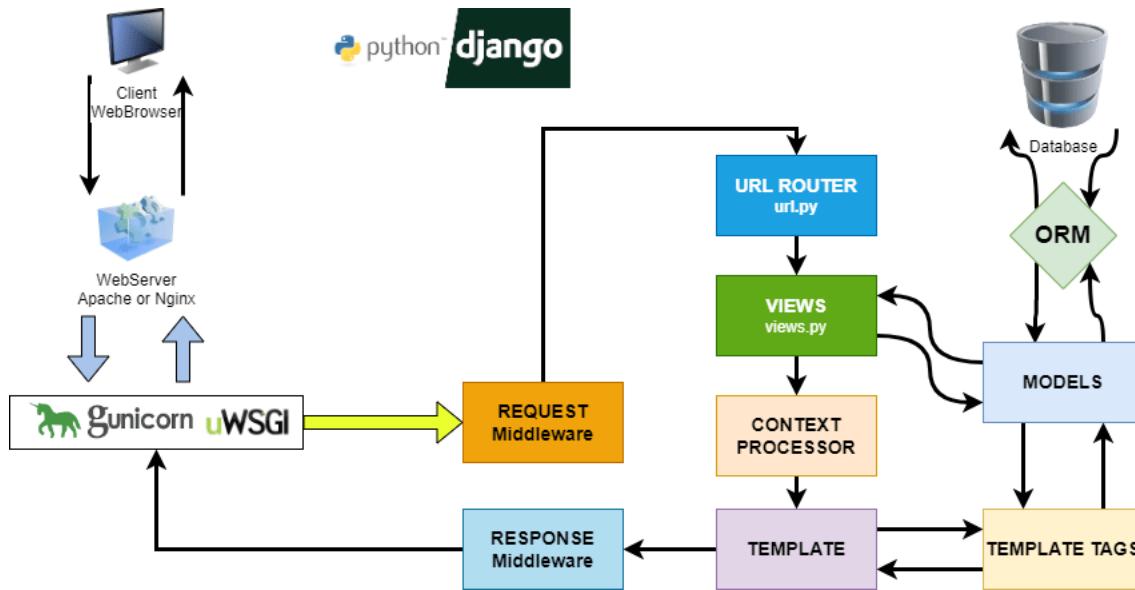
When a user requests a URL, Django decides which view it will send it to.

This is done in a file called **urls.py**

The MVT pattern is specified in the Django framework as follows:

1. The user requests a resource.
2. Django receives the URL, checks the **urls.py** file, and calls the view that matches the URL.

3. The view, located in `views.py`, checks for relevant models.
4. The models are imported from the `models.py` file.
5. The view then sends the data to a specified template in the `template` folder.
6. The template contains HTML and Django tags, and with the data it returns finished HTML content back to the browser.



While Django takes care of the Controller part itself (the software code that controls the interactions between the Model and View), the developer is freed up to focus on the Model, View and Template parts. This is why the Django system is referred to as an MVT system, rather than MVC.

2.0.2 Components of Django

1. **Django ORM (Object-Relational Mapping):** Django ORM allows developers to interact with their database, like they would with SQL. In other words, it's a way to create, retrieve, update and delete records from your database using Python.
2. **Models:** Models are a single, definitive source of information about your data. It contains the essential fields and behaviors of the data you're storing. Generally, each model maps to a single database table.

3. **Views:** A view function, or view for short, is a Python function that takes a Web request and returns a Web response. This response can be the HTML contents of a Web page, or a redirect, or a 404 error, or an XML document, or an image... or anything, really.
4. **Templates:** Django's template language is designed to strike a balance between power and ease. It's designed to feel comfortable to those used to working with HTML.
5. **Forms:** Handling forms is a complex business. Forms are a flexible mechanism for collecting user input because there are suitable widgets for entering many different types of data, including text boxes, checkboxes, radio buttons, date pickers and so on.
6. **Admin Interface:** Django includes a framework for creating admin interfaces. This is a very powerful feature, and one that makes it quick and easy to start actually using Django for real projects.
7. **URL Dispatcher:** A clean, elegant URL scheme is an important detail in a high-quality Web application. Django lets you design URLs however you want, with no framework limitations.
8. **Middleware:** Middleware is a series of hooks into Django's request/response processing. It's a light, low-level "plugin" system for globally altering Django's input or output.
9. **Authentication:** Django comes with a user authentication system. It handles user accounts, groups, permissions and cookie-based user sessions.
10. **Internationalization:** Django offers full support for translating text into different languages, plus locale-specific formatting of dates, times, numbers, and time zones.
11. **Testing tools:** Django provides a small set of tools that come in handy when writing tests.
12. **Asynchronous Tasks:** request-response flow, enabling certain types of operations to continue to run after a response has been sent back to the client, or to be scheduled to run at a later time.

(a) **Asynchronous Views and Middleware (Async Views):**

Starting with Django 3.1, Django began to support asynchronous ("async") views natively. An async view is just like a regular view, but it is declared with "async def" instead of "def". An asynchronous view can perform operations that would block if run synchronously (like making HTTP requests to a remote API, reading a file from the disk, etc.) without blocking the execution of the rest of the server.

(b) **Task Queues (like Celery):** Task queues allow you to perform tasks outside the scope of a request-response cycle, as well as schedule tasks to run at certain intervals. A task queue will store tasks until a worker process is ready to execute them. The most common task queue used with Django is Celery, though others like Huey, RQ (Redis Queue), etc. can also be used.

13. **Serialization:** Django's serialization framework provides a mechanism for "translating" Django models into other formats. Usually these other formats will be text-based and used for sending Django data over a wire, but it's possible for a serializer to handle any format (text-based or not).
14. **Caching:** Django comes with a robust cache system that lets you save dynamic pages so they don't have to be calculated for each request. For convenience, Django offers different levels of cache granularity: You can cache the output of specific views, you can cache only the pieces that are difficult to produce, or you can cache your entire site.
15. **Signals:** Django includes a "signal dispatcher" which helps decoupled applications get notified when actions occur elsewhere in the framework. In a nutshell, signals allow certain senders to notify a set of receivers that some action has taken place. They're especially useful when many pieces of code may be interested in the same events.



Django Install

3.1 ➤ prerequisite

3.1.1 ➤ Python

Django requires Python. Different versions of Django require different versions. However, they are backward compatible with the latest version of Python.

Check that Python is installed.

```
██████████ occc@occc-sunyorange
occc@occc-VirtualBox: $ python3 --version
Python 3.10.6
occc@occc-VirtualBox: $
```

3.1.2 ➤ PIP

```
occc@occc-sunyorange
occc@occc-VirtualBox: $ pip –version
Command 'pip' not found, but can be installed with:
sudo apt install python3-pip
occc@occc-VirtualBox: $
```

If pip is not installed then run:

```
occc@occc-sunyorange
occc@occc-VirtualBox: $ sudo apt install python3-pip
[sudo] password for occc:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
build-essential dpkg-dev fakeroot g++ g++-11 javascript-common
libalgorithm-diff-perl libalgorithm-diff-xs-perl libalgorithm-merge-perl
libdpkg-perl libexpat1-dev libfakeroot libfile-fcntllock-perl libjs-jquery
libjs-sphinxdoc libjs-underscore libpython3-dev libpython3.10-dev
libstdc++-11-dev lto-disabled-list python3-dev python3-distutils
python3-setuptools python3-wheel python3.10-dev zlib1g-dev
Suggested packages:
debian-keyring g++-multilib g++-11-multilib gcc-11-doc git bzr
libstdc++-11-doc python-setuptools-doc
The following NEW packages will be installed:
build-essential dpkg-dev fakeroot g++ g++-11 javascript-common
libalgorithm-diff-perl libalgorithm-diff-xs-perl libalgorithm-merge-perl
libdpkg-perl libexpat1-dev libfakeroot libfile-fcntllock-perl libjs-jquery
libjs-sphinxdoc libjs-underscore libpython3-dev libpython3.10-dev
libstdc++-11-dev lto-disabled-list python3-dev python3-distutils python3-pip
python3-setuptools python3-wheel python3.10-dev zlib1g-dev
0 upgraded, 27 newly installed, 0 to remove and 0 not upgraded.
Need to get 22.9 MB of archives.
After this operation, 88.6 MB of additional disk space will be used.
Do you want to continue? [Y/n] y
```

```
occc@occc-sunyorange
occc@occc-VirtualBox: $ pip –version
pip 22.0.2 from /usr/lib/python3/dist-packages/pip (python 3.10)
occc@occc-VirtualBox: $
```

3.1.3 create a project directory

```
occc@occc-VirtualBox:/$ sudo mkdir dj  
occc@occc-VirtualBox:/$ sudo chown occc dj  
occc@occc-VirtualBox:/$ sudo chown occc dj  
  
occc@occc-VirtualBox:/dj$ python3 -m venv orange  
The virtual environment was not created successfully because ensurepip is  
not  
available. On Debian/Ubuntu systems, you need to install the python3-venv  
package using the following command.
```

apt install python3.10-venv

You may need to use sudo with that command. After installing the python3-venv package, recreate your virtual environment.

```
occc@occc-VirtualBox:/dj$
```

3.1.4 Virtual Environment

It is suggested to have a dedicated virtual environment for each Django project, and one way to manage a virtual environment is **venv**, which is part of Python but not always installed.

The name of the virtual environment is your choice; in this tutorial we will call it **venv**.

Type the following in the command prompt; remember to navigate to where you want to create your project:



occc@occc-sunyorange

occc@occc-VirtualBox: \$ cd dj

occc@occc-VirtualBox:/dj\$ python3 -m venv venv

The virtual environment was not created successfully because ensurepip is not available. On Debian/Ubuntu systems, you need to install the python3-venv package using the following command.

apt install python3.10-venv

You may need to use sudo with that command. After installing the python3-venv package, recreate your virtual environment.

occc@occc-VirtualBox:/dj\$

3.1.5 Install venv



occc@occc-sunyorange

occc@occc-VirtualBox:/dj\$ sudo apt install python3.10-venv

Reading package lists... Done

Building dependency tree... Done

Reading state information... Done

The following additional packages will be installed:

python3-pip-whl python3-setuptools-whl

The following NEW packages will be installed:

python3-pip-whl python3-setuptools-whl python3.10-venv

0 upgraded, 3 newly installed, 0 to remove and 0 not upgraded.

Need to get 2,473 kB of archives.

After this operation, 2,882 kB of additional disk space will be used.

Do you want to continue? [Y/n] y

occc@occc-VirtualBox:/dj\$

3.1.6 create venv project



occc@occc-sunyorange

occc@occc-VirtualBox:/dj\$ python3 -m venv venv

occc@occc-VirtualBox:/dj\$

The above command will set up a virtual environment and create a folder named "venv" with subfolders and files, like this:

```
occc@occc-VirtualBox:/dj$ cd venv  
  
occc@occc-VirtualBox:/dj/venv$ ls -l  
total 16  
drwxrwxr-x 2 occc occc 4096 Jun 11 13:33 bin  
drwxrwxr-x 2 occc occc 4096 Jun 11 13:27 include  
drwxrwxr-x 3 occc occc 4096 Jun 11 13:27 lib  
lrwxrwxrwx 1 occc occc 3 Jun 11 13:27 lib64 -> lib  
-rw-rw-r-- 1 occc occc 70 Jun 11 13:33 pyvenv.cfg  
occc@occc-VirtualBox:/dj/venv$
```

You have to activate this new virtual environment by typing this command:

```
occc@occc-VirtualBox:/dj$ pwd  
/dj  
occc@occc-VirtualBox:/dj$ source venv/bin/activate  
(venv) occc@occc-VirtualBox:/dj$
```

Note: You must activate the virtual environment whenever you open the command prompt to work on your project.

3.2 > Install Django

Note: Remember to install Django while in the virtual environment, and it is activated!

```
occc@occc-VirtualBox:/dj$ source venv/bin/activate
(venv) occc@occc-VirtualBox:/dj$ python -m pip install django
Collecting django
  Using cached Django-4.2.3-py3-none-any.whl (8.0 MB)
Requirement already satisfied: asgiref<4,>=3.6.0 in ./venv/lib/python3.10/site-packages (from django) (3.7.2)
Requirement already satisfied: sqlparse>=0.3.1 in ./venv/lib/python3.10/site-packages (from django) (0.4.4)
Requirement already satisfied: typing-extensions>=4 in ./venv/lib/python3.10/site-packages (from asgiref<4,>=3.6.0->django) (4.7.1)
Installing collected packages: django
Successfully installed django-4.2.3
(venv) occc@occc-VirtualBox:/dj$
```

3.2.1 Check Django Version



occc@occc-sunyorange

```
(venv) occc@occc-VirtualBox:/dj$ django-admin --version
4.2.3
(venv) occc@occc-VirtualBox:/dj$
```

3.3 First project



occc@occc-sunyorange

```
()venv) occc@occc-VirtualBox:/dj$ ls
venv
(venv) occc@occc-VirtualBox:/dj$
```

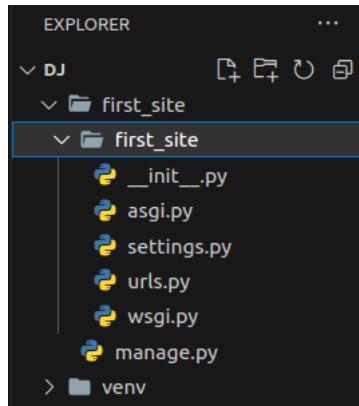
Create a project by issuing: **django-admin startproject first-site**



occc@occc-sunyorange

```
(venv) occc@occc-VirtualBox:/dj$ django-admin startproject first_site
(venv) occc@occc-VirtualBox:/dj$ ls
first_site venv
(venv) occc@occc-VirtualBox:/dj$
```

Django creates a `first_site` folder on your computer with this content:



Django will have several files, and most Django projects will have the following structure:

Files from the above output:

- The outer `first_site/` root directory is a container for your project. Its name doesn't matter to Django; you can rename it to anything you like.
- `first_site/__init__.py`: An empty file that tells Python that this directory should be considered a Python package. If you're a Python beginner, read more about packages in the [official Python docs](#).
- `first_site/settings.py`: Settings/configuration for this Django project. [Django settings](#) will tell you all about how settings work.
- `first_site/urls.py`: The URL declarations for this Django project; a “table of contents” of your Django-powered site. You can read more about URLs in [URL dispatcher](#) .
- `manage.py` A command-line utility that lets you interact with this Django project in various ways. You can read all the details about `manage.py` in [django-admin and manage.py](#).
- The inner `first_site/` directory is the actual Python package for your project. Its name is the Python package name you'll need to use to import anything inside it (e.g. `first_site.urls`).
- The inner `first_site/asgi.py`: An entry-point for ASGI-compatible web servers to serve your project. See [How to deploy with ASGI](#)

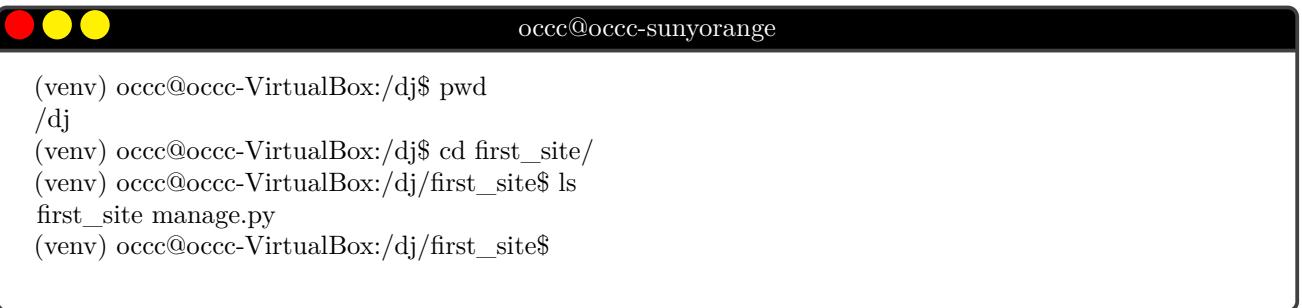
for more details.

- The inner `first_site//wsgi.py`: An entry-point for WSGI-compatible web servers to serve your project. See [How to deploy with WSGI](#) for more details.

3.4 ➤ test if our project is working

Django comes with a simple web server pre-installed. It's very convenient during the development, so we don't have to install anything else to run the project locally.

Navigate to our `first_site` project

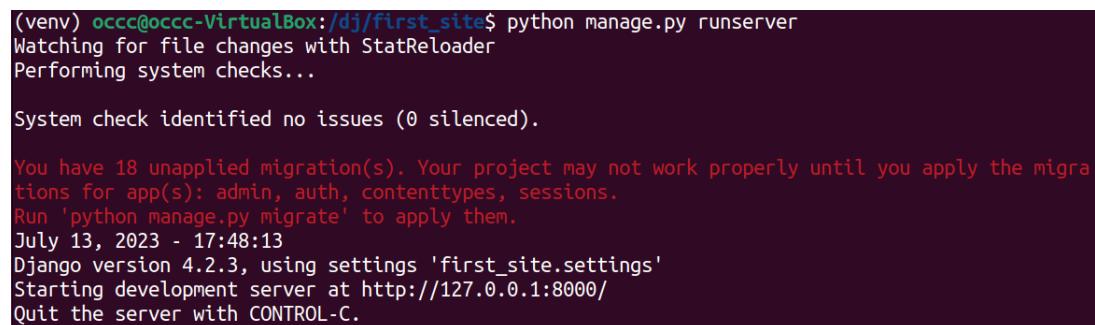


```
(venv) occc@occc-VirtualBox:/dj$ pwd
/dj
(venv) occc@occc-VirtualBox:/dj$ cd first_site/
(venv) occc@occc-VirtualBox:/dj/first_site$ ls
first_site manage.py
(venv) occc@occc-VirtualBox:/dj/first_site$
```

We can test that our installation is working by executing the following command:



```
python manage.py runserver
```

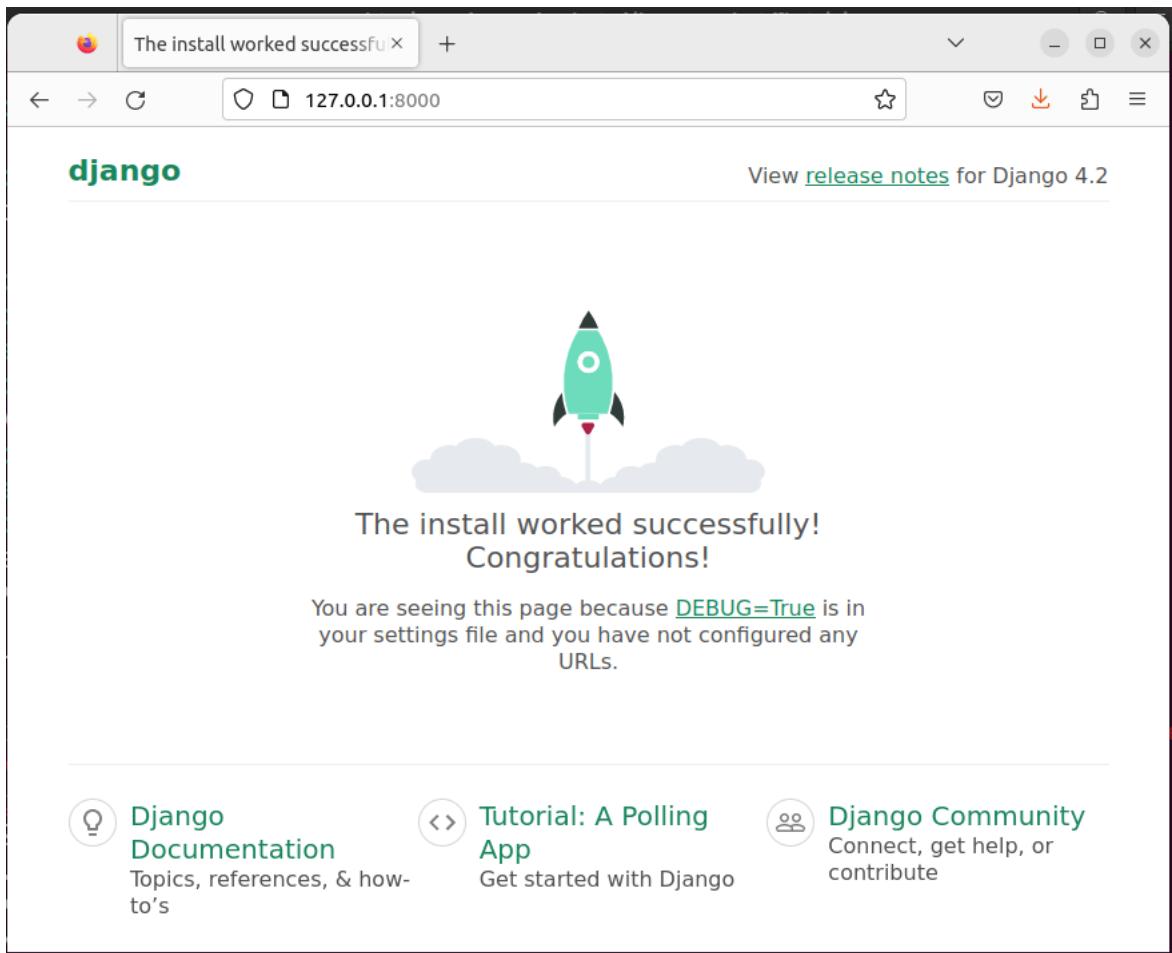


```
(venv) occc@occc-VirtualBox:/dj/first_site$ python manage.py runserver
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).

You have 18 unapplied migration(s). Your project may not work properly until you apply the migrations for app(s): admin, auth, contenttypes, sessions.
Run 'python manage.py migrate' to apply them.
July 13, 2023 - 17:48:13
Django version 4.2.3, using settings 'first_site.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CONTROL-C.
```

Ignore the errors for now. Use a web browser to navigate to the URL specified on the command line output.



Type in the **CONTROL-C** sequence to stop the server.

Changing the port

By default, the runserver command starts the development server on the internal IP at port 8000.

If you want to change the server's port, pass it as a command-line argument. For instance, this command starts the server on port 8080:

```
$ python manage.py runserver 8080
```

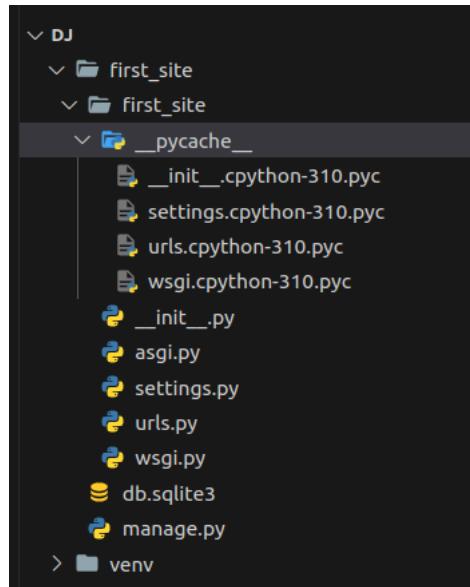
If you want to change the server's IP, pass it along with the port. For example, to listen on all available public IPs (which is useful if you are running Vagrant or want to show off your work on other computers on the network), use:

```
$ python manage.py runserver 0.0.0.0:8000
```

Full docs for the development server can be found in the [runserver](#) reference.

Project

A **project** is a web application using Django. There is only ever one project and many “**apps**” within it.



Note that several new files were created when we started our development web server.

INSTALLED_APPS

Within the newly created **settings.py** file is a configuration called **INSTALLED_APPS**, a list of Django apps within a project. Django comes with six built-in apps that we can examine.

```
1 """
2 """
3 Django settings for first_site project.
4
5 Generated by 'django-admin startproject' using Django 4.2.3.
6
7 For more information on this file, see
8 https://docs.djangoproject.com/en/4.2/topics/settings/
9
10 For the full list of settings and their values, see
11 https://docs.djangoproject.com/en/4.2/ref/settings/
12 """
13
```

```
14 from pathlib import Path
15
16 # Build paths inside the project like this: BASE_DIR / 'subdir'.
17 BASE_DIR = Path(__file__).resolve().parent.parent
18
19
20 # Quick-start development settings - unsuitable for production
21 # See https://docs.djangoproject.com/en/4.2/howto/deployment/checklist/
22
23 # SECURITY WARNING: keep the secret key used in production secret!
24 SECRET_KEY = 'django-insecure-ip@@#k&zrk-zn6^v-md4=c_*=wi185+f_(##&#z75_*'
25     ↴ vvfyu!m4#'
26
27 # SECURITY WARNING: don't run with debug turned on in production!
28 DEBUG = True
29
30
31
32 # Application definition
33
34 INSTALLED_APPS = [
35     'django.contrib.admin',
36     'django.contrib.auth',
37     'django.contrib.contenttypes',
38     'django.contrib.sessions',
39     'django.contrib.messages',
40     'django.contrib.staticfiles',
41 ]
42
43 MIDDLEWARE = [
44     'django.middleware.security.SecurityMiddleware',
45     'django.contrib.sessions.middleware.SessionMiddleware',
46     'django.middleware.common.CommonMiddleware',
47     'django.middleware.csrf.CsrfViewMiddleware',
48     'django.contrib.auth.middleware.AuthenticationMiddleware',
49     'django.contrib.messages.middleware.MessageMiddleware',
50     'django.middleware.clickjacking.XFrameOptionsMiddleware',
51 ]
52
53 ROOT_URLCONF = 'first_site.urls'
54
55 TEMPLATES = [
56     {
57         'BACKEND': 'django.template.backends.django.DjangoTemplates',
58         'DIRS': [],
59         'APP_DIRS': True,
60         'OPTIONS': {
61             'context_processors': [
62                 'django.template.context_processors.debug',
63                 'django.template.context_processors.request',
64                 'django.contrib.auth.context_processors.auth',
65                 'django.contrib.messages.context_processors.messages',
66             ],
67         },
68     },
69 ]
```

```
70 WSGI_APPLICATION = 'first_site.wsgi.application'
71
72
73
74 # Database
75 # https://docs.djangoproject.com/en/4.2/ref/settings/#databases
76
77 DATABASES = {
78     'default': {
79         'ENGINE': 'django.db.backends.sqlite3',
80         'NAME': BASE_DIR / 'db.sqlite3',
81     }
82 }
83
84
85 # Password validation
86 # https://docs.djangoproject.com/en/4.2/ref/settings/#auth-password-
87     ↴ validators
88
89 AUTH_PASSWORD_VALIDATORS = [
90     {
91         'NAME': 'django.contrib.auth.password_validation.(
92             ↴ UserAttributeSimilarityValidator',
93     },
94     {
95         'NAME': 'django.contrib.auth.password_validation.(
96             ↴ MinimumLengthValidator',
97     },
98     {
99         'NAME': 'django.contrib.auth.password_validation.(
100            ↴ CommonPasswordValidator',
101    },
102    {
103        'NAME': 'django.contrib.auth.password_validation.(
104            ↴ NumericPasswordValidator',
105    },
106 ]
107
108
109 # Internationalization
110 # https://docs.djangoproject.com/en/4.2/topics/i18n/
111 LANGUAGE_CODE = 'en-us'
112
113 TIME_ZONE = 'UTC'
114
115 USE_I18N = True
116
117 USE_TZ = True
118
119 # Static files (CSS, JavaScript, Images)
120 # https://docs.djangoproject.com/en/4.2/howto/static-files/
121 STATIC_URL = 'static/'
122
123 # Default primary key field type
```

```
122 # https://docs.djangoproject.com/en/4.2/ref/settings/#default-auto-field
123
124 DEFAULT_AUTO_FIELD = 'django.db.models.BigAutoField'
```

We are interested in one part of the file in particular.

```
# Application definition

INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
]
```

These are the six built-in apps created during the project setup.

- **django.contrib.admin** The admin site.
- **django.contrib.auth** An authentication system.
- **django.contrib.contenttypes** A framework for content types.
- **django.contrib.sessions** A session framework.
- **django.contrib.messages** A messaging framework.
- **django.contrib.staticfiles** A framework for managing static files.

Apps

In Django, an "app" is a self-contained module designed to provide specific functionality within a larger Django project. An app could be anything from a user authentication system to a blog, a forum, a shopping cart, or an API. Each Django app supplies some functionality encapsulated as its own Python package that can be dropped into any Django project.

- **Modularity:** Django apps are designed to be plug-and-play, meaning you can take an app from one project and drop it into another, and it should work with minimal setup.
- **Self-Contained:** A Django app is self-contained and should not rely on other apps. The exception is Django's built-in apps like `django.contrib.auth` that you can use for user authentication, groups, permissions, and sessions.
- **Reuse:** Apps can be reused across projects. For example, if you build a Django app to handle user authentication, you could potentially use this in another project without having to write the functionality again.
- **Structure:** A Django app typically includes models (for database interaction), views (for handling HTTP requests and responses), and templates (for defining the structure of the HTML that the app returns), though not all apps will require all three.
- **App Registry:** Django keeps track of all installed applications in the `INSTALLED_APPS` setting of the project configuration file. Only apps that are included in this list can be used by the Django project.

In Django, you use the command `python manage.py startapp <appname>` to create a new application. This command will create a directory with the necessary files and structure for the new application.

Overall, the concept of apps in Django allows for a high degree of modularity and reusability, making Django a powerful tool for web application development.

Let's add the `csc108` app now. When you create projects and apps in Django, apps will be created inside the main project folder, at the same level as the project's settings folder (usually it's the folder with the same name as your project).

Go to the directory where the `manage.py` file is and execute the following command:

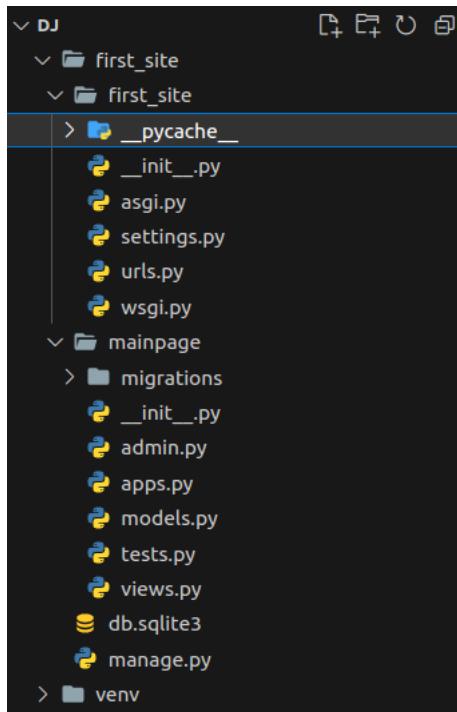


```
occc@occc-sunyorange: ~
```

```
django-admin startapp mainpage
```

```
(venv) occc@occc-VirtualBox:/dj/first_site$ ls
db.sqlite3 first_site manage.py
(venv) occc@occc-VirtualBox:/dj/first_site$ pwd
/dj/first_site
(venv) occc@occc-VirtualBox:/dj/first_site$ ls
db.sqlite3 first_site manage.py
(venv) occc@occc-VirtualBox:/dj/first_site$ django-admin startapp mainpage
(venv) occc@occc-VirtualBox:/dj/first_site$
```

Here is our complete updated structure.



Let's first explore what each file does:

- **migrations/** here Django stores some files to keep track of the changes you create in the **models.py** file, in order to keep the database and the **models.py** synchronized.
- **admin.py** file is where you define the administrative interface for your app. Django comes with a built-in admin site that's intended for site administrators. It's a very powerful feature and can save you a lot of time because it's fully functional and ready to be used out of the box.
- **apps.py** The apps.py file in Django is a configuration file for a specific Django app. It helps Django understand and configure certain aspects

of the app.

Here's an example of what a simple apps.py file might look like:

apps.py example

```
1  from django.apps import AppConfig  
2  
3  class AppConfig(AppConfig):  
4      default_auto_field = 'django.db.models.BigAutoField'  
5      name = 'myapp'
```

The `name` attribute is a human-readable name for the app that can be used in administrative interfaces. It must match the name of the Python package that the application lives in.

The `default_auto_field` attribute is new since Django 3.2. It defines the type of auto-created primary key to use on models which do not explicitly define a field named 'id'. By default, Django uses 'AutoField', which is an integer field. However, you may want to use a different auto-field like 'BigAutoField' if you expect the database to grow rapidly.

The AppConfig class can also have other attributes and methods. For example:

- `verbose_name`: A human-readable name for the app.
- `path`: A Python filesystem path to the application directory.
- `label`: A short, unique, and URL-friendly name for the application.
- `ready() method`: A method that is called at the end of Django's startup sequence. You can override this method to perform app-specific tasks, such as registering signals, models, or anything else that needs to be ready for the application.

The apps.py file is automatically created when you start a new app with Django's `startapp` command, but you may not need to modify it for simple applications. The apps.py file can be very important for customizing your app's behavior for larger and more complex applications.

- **tests.py** file is where you write tests for your application to ensure it works as expected. Testing is vital to software development; that helps you catch errors and avoid regressions.

Django provides a built-in testing framework that you can use to simulate requests, insert test data, inspect the application's output, and generally ensure your code is doing what it should.

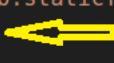
- **views.py** file is where you define the views for your application. Views in Django are a key component of the MVC (Model-View-Controller) architecture, which Django refers to as the MVT (Model-View-Template) architecture.

A view in Django is a Python function (or class, in the case of class-based views) that receives a web request and returns a web response. The response can be anything - HTML content, a redirect, a 404 error, an XML document, an image, or anything else.

Now that we created our first app, let's configure our project to use it.

To do that, open the **settings.py** and find the **INSTALLED_APPS** variable:

Add our mainpage app to the list of **INSTALLED_APPS**:

```
INSTALLED_APPS = [  
    'django.contrib.admin',  
    'django.contrib.auth',  
    'django.contrib.contenttypes',  
    'django.contrib.sessions',  
    'django.contrib.messages',  
    'django.contrib.staticfiles',  
    'mainpage', ]
```

Hello world from mainpage

3.5.1 > Views

Write our first view

Open the file `mainpage/views.py` and put the following Python code in it:

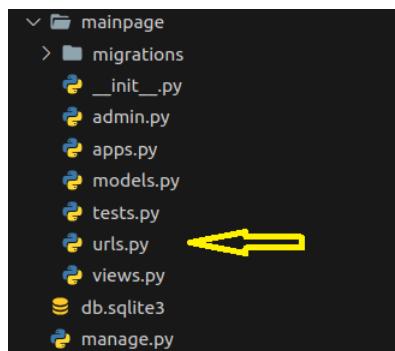
`mainpage/views.py`

```

1 from django.shortcuts import render
2 #import for direct httpResponse
3 from django.http import HttpResponseRedirect
4
5
6 def index(request):
7     return HttpResponseRedirect("Hello world from mainpage of csc108 webprogramming ↴
8     ↴ 1.")
# Create your views here.
```

The above code is the most straightforward view possible in Django. To call the view, we need to map it to a URL - and for this, we need a **URLconf**.

To create a URLconf in the mainpage directory, create a file called `urls.py`. Your mainpage app directory should now look like this:



In the `mainpage/urls.py` file, include the following code:

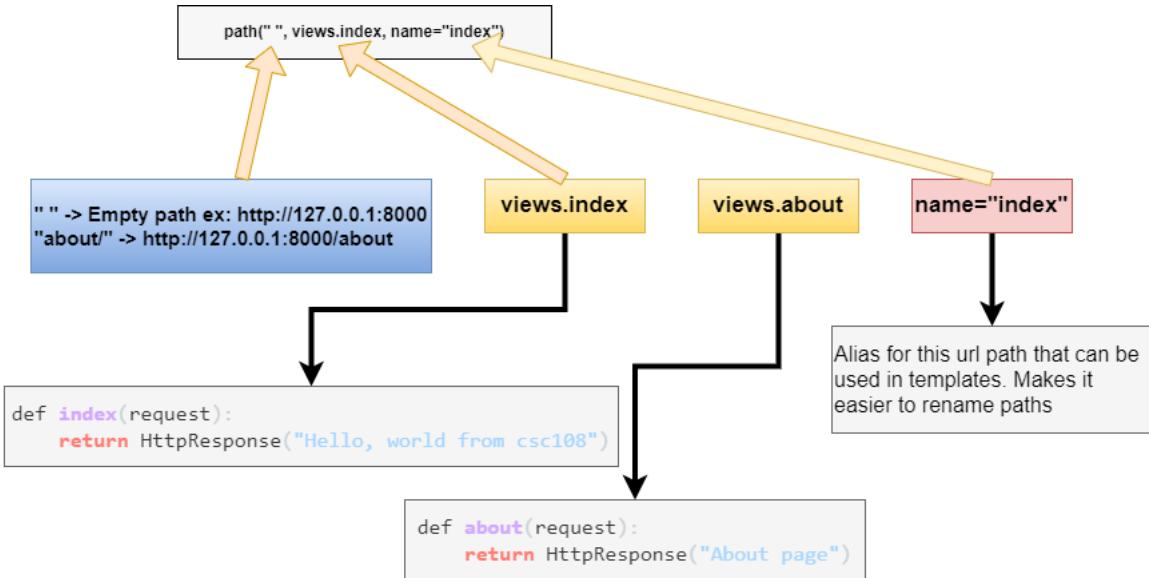
`mainpage/urls.py`

```

1 from django.urls import path
2
3 from . import views
4
5 urlpatterns = [
6     path("", views.index, name="index"),
```

7]

URL documentation



The next step is to point the root URLconf at the mainpage.urls module. In first_site/urls.py, add an import for django.urls include and path then insert and include() in the urlpatterns list, so you have:

project urls.py

```

1 """
2 URL configuration for first_lab project.
3
4 The `urlpatterns` list routes URLs to views. For more information please see:
5     https://docs.djangoproject.com/en/4.2/topics/http/urls/
6 Examples:
7 Function views
8 1. Add an import: from my_app import views
9 2. Add a URL to urlpatterns: path('', views.home, name='home')
10 Class-based views
11 1. Add an import: from other_app.views import Home
12 2. Add a URL to urlpatterns: path('', Home.as_view(), name='home')
13 Including another URLconf
14 1. Import the include() function: from django.urls import include, path
15 2. Add a URL to urlpatterns: path('blog/', include('blog.urls'))
16 """
17 from django.contrib import admin
18 from django.urls import include, path
19
20 urlpatterns = [
21     path("csc108/", include("mainpage.urls")),
22     path('admin/', admin.site.urls),

```

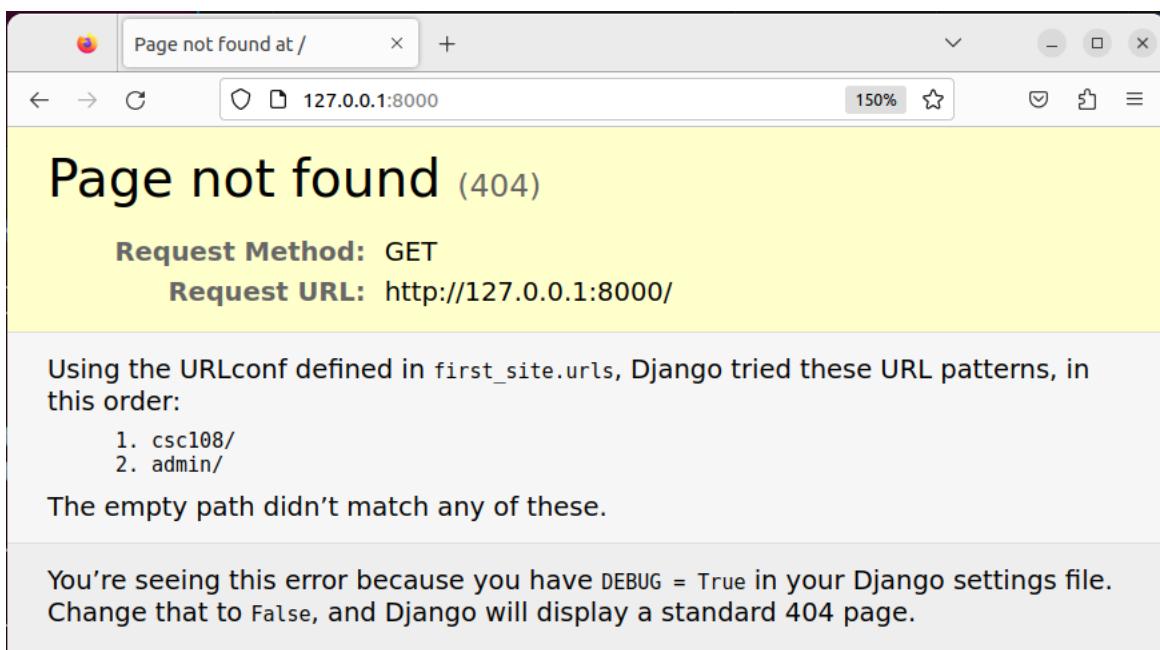
23]

The `include()` function allows referencing other URLconfs. Whenever Django encounters `include()`, it chops off whatever part of the URL matched up to that point and sends the remaining string to the included URLconf for further processing.

The idea behind `include()` is to make it easy to plug-and-play URLs. Since `mainpage` has its own URLconf (`mainpage/urls.py`), they can be placed under “`/mainpage/`”, or under “`/webprog1/`”, or under “`/content/csc108/`”, or any other path root, and the app will still work.



If you get an error, make sure you are pointing to the `csc108` view and not the main one as before. The built-in main website is no longer active since there is no empty path in the main `urls.py`.



The `path()` function is passed four arguments, two required: route and view,

and two optional: `kwargs`, and `name`.

path() argument: route

`route` is a string that contains a URL pattern. When processing a request, Django starts at the first pattern in `urlpatterns` and makes its way down the list, comparing the requested URL against each pattern until it finds one that matches.

Patterns don't search GET and POST parameters, or the domain name. For example, in a request to `https://www.example.com/myapp/`, the `URLconf` will look for `myapp/`. In a request to `https://www.example.com/myapp/?page=3`, the `URLconf` will also look for `myapp/`.

path() argument: view

When Django finds a matching pattern, it calls the specified view function with an `HttpRequest` object as the first argument and any "captured" values from the route as keyword arguments. This will be covered more later.

path() argument: kwargs

Arbitrary keyword arguments can be passed in a dictionary to the target view.

path() argument: name

Naming your URL lets you refer to it unambiguously from elsewhere in Django, especially from within templates. This powerful feature allows you to make global changes to the URL patterns of your project while only touching a single file.

There are a couple of items that we have to take care of before our basic install is done and tested.

Our hosts file has the following:

```
127.0.0.1 localhost csc108.sunyorange.edu.local
127.0.1.1 occc-VirtualBox
```

```
# The following lines are desirable for IPv6 capable hosts
::1 ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
```

However even though we can resolve csc108.sunyorange.edu.local django will not allow us to display any pages.

DisallowHost at /csc108

Invalid HTTP_HOST header: 'csc108.sunyorange.edu.local:8000'. You may need to add 'csc108.sunyorange.edu.local' to ALLOWED_HOSTS.

Request Method: GET
Request URL: http://csc108.sunyorange.edu.local:8000/csc108
Django Version: 4.2.3
Exception Type: DisallowedHost
Exception Value: Invalid HTTP_HOST header: 'csc108.sunyorange.edu.local:8000'. You may need to add 'csc108.sunyorange.edu.local' to ALLOWED_HOSTS.
Exception Location: /dj/venv/lib/python3.10/site-packages/django/http/request.py, line 150, in get_host
Python Executable: /dj/venv/bin/python
Python Version: 3.10.6
Python Path: ['/dj/first_site', '/usr/lib/python310.zip', '/usr/lib/python3.10', '/usr/lib/python3.10/lib-dynload', '/dj/venv/lib/python3.10/site-packages']
Server time: Thu, 13 Jul 2023 18:49:51 +0000

Traceback [Switch to copy-and-paste view](#)

```
/dj/venv/lib/python3.10/site-packages/django/core/handlers/exception.py, line 55, in inner
    55.         response = get_response(request)
▶ Local vars
/dj/venv/lib/python3.10/site-packages/django/utils/deprecation.py, line 133, in __call__
    133.     response = self.process_request(request)
```

From the command line where we started our runserver we can also see that there were errors.

```
raise DisallowedHost(msg)
django.core.exceptions.DisallowedHost: Invalid HTTP_HOST header: 'csc108.sunyorange.edu.local:8000'. You may need to add 'csc108.sunyorange.edu.local' to ALLOWED_HOSTS.
Bad Request: /csc108
[13/Jul/2023 18:49:51] "GET /csc108 HTTP/1.1" 400 71242
Invalid HTTP_HOST header: 'csc108.sunyorange.edu.local:8000'. You may need to add 'csc108.sunyorange.edu.local' to ALLOWED_HOSTS.
Traceback (most recent call last):
  File "/dj/venv/lib/python3.10/site-packages/django/core/handlers/exception.py", line 5
5, in inner
    response = get_response(request)
  File "/dj/venv/lib/python3.10/site-packages/django/utils/deprecation.py", line 133, in
__call__
    response = self.process_request(request)
  File "/dj/venv/lib/python3.10/site-packages/django/middleware/common.py", line 48, in
process_request
    host = request.get_host()
  File "/dj/venv/lib/python3.10/site-packages/django/http/request.py", line 150, in get_
host
    raise DisallowedHost(msg)
django.core.exceptions.DisallowedHost: Invalid HTTP_HOST header: 'csc108.sunyorange.edu.local:8000'. You may need to add 'csc108.sunyorange.edu.local' to ALLOWED_HOSTS.
Bad Request: /favicon.ico
```

Edit the project `settings.py` file and add the domain names for this server.

```
# SECURITY WARNING: don't run with debug turned on in production!
DEBUG = True

ALLOWED_HOSTS = ['csc108.sunyorange.edu.local']
```

In production, disable the DEBUG option for security.



Add database

Almost all large websites will use a database as a back end to store information and to dynamically create web pages.

[Django DB documentation](#)

Django officially supports the following databases:

- PostgreSQL
- MariaDB
- MySQL
- Oracle
- SQLite

Why setup a database early in the project? Django always expects some database to be present. We saw this when we started our webserver and had errors.

```
^C(venv) occc@occc-VirtualBox:/dj/first_site$ python manage.py runserver
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).

You have 18 unapplied migration(s). Your project may not work properly until you apply t
he migrations for app(s): admin, auth, contenttypes, sessions.
Run 'python manage.py migrate' to apply them.
July 13, 2023 - 18:57:47
Django version 4.2.3, using settings 'first_site.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CONTROL-C.
```

The errors in red were from un-applied database changes.

By default, Django is setup with sqlite3 database since it comes with Python. This database is a simple file-based database that is not suitable for any large-scale deployment.

In our example, we will use PostgreSQL as our RDBMS.

4.1 ➤ django postgresql setup

run sudo apt update

```
● ● ● occc@occc-sunyorange
(orange) occc@occc-VirtualBox:/dj$ sudo apt update
[sudo] password for occc:
```

Install new software packages

```
● ● ● occc@occc-sunyorange
sudo apt install python3-pip python3-dev libpq-dev postgresql postgresql-contrib nginx curl
```

We installed some additional software, such as nginx.

4.1.1

Creating the PostgreSQL Database and Users

By default, Postgres uses a “peer authentication” scheme for local connections. This means that if the user’s operating system username matches a valid Postgres username, that user can log in without further authentication.

During the Postgres installation, an operating system user named **postgres** was created to correspond to the **postgres** PostgreSQL administrative user. We need to use this user to perform administrative tasks. We can use sudo and pass in the username with the **-u** option.

Log into an interactive Postgres session by typing:



```
occc@occc-sunyorange
sudo -u postgres psql

(venv) occc@occc-VirtualBox:/dj/first_site$ sudo -u postgres psql
[sudo] password for occc:
psql (14.8 (Ubuntu 14.8-0ubuntu0.22.04.1))
Type "help" for help.

postgres=#
```

Create a database for your project:

note

Every Postgres statement must end with a semi-colon, so make sure that your command ends with one if you are experiencing issues.

```
postgres=# CREATE DATABASE orange;
CREATE DATABASE
postgres=#

```

Create a database user for our project. Make sure to select a secure password:

```
CREATE USER orangeuser WITH PASSWORD 'orange123';
```

<https://www.postgresql.org/docs/15/sql-createuser.html>

```
postgres=# CREATE USER orangeuser WITH PASSWORD 'orange123';
CREATE ROLE
postgres=#
```

We must modify a few connection parameters for the user we just created. This will speed up database operations so that the correct values do not have to be queried and set each time a connection is established.

We are setting the default encoding to **UTF-8**, which Django expects. We are also setting the default transaction isolation scheme to “read committed”, which blocks reads from uncommitted transactions. Lastly, we are setting the timezone. By default, our Django projects will be set to use **UTC**. These are all recommendations from the Django project itself:

[django optimizing-postgresql-s-configuration](#)

```
ALTER ROLE orangeuser SET client_encoding TO 'utf8';
ALTER ROLE orangeuser SET default_transaction_isolation TO 'read committed';
ALTER ROLE orangeuser SET timezone TO 'UTC';
```

```
postgres=# ALTER ROLE orangeuser SET client_encoding TO 'utf8';
ALTER ROLE
postgres=# ALTER ROLE orangeuser SET default_transaction_isolation TO 'read committed';
ALTER ROLE
postgres=# ALTER ROLE orangeuser SET timezone TO 'UTC';
ALTER ROLE
postgres=#
```

Now, we can give our new user access to administer our new database:

```
GRANT ALL PRIVILEGES ON DATABASE orange TO orangeuser;
```

```
postgres=# GRANT ALL PRIVILEGES ON DATABASE orange TO orangeuser;
GRANT
postgres=#
```

List all Users in PostgreSQL

use the \du or \du+ commands to verify that the user is there.

```
postgres=# \du
          List of roles
Role name | Attributes | Member of
-----+-----+-----+
orangeuser |
postgres   | Superuser, Create role, Create DB, Replication, Bypass RLS | {}

postgres=# \du+
          List of roles
Role name | Attributes | Member of | Description
-----+-----+-----+-----+
orangeuser |
postgres   | Superuser, Create role, Create DB, Replication, Bypass RLS | {} |
```

When you are finished, exit out of the PostgreSQL prompt by typing: `\q`

4.1.2 Install database bindings



occc@occc-sunyorange

```
pip install gunicorn psycopg2-binary
```

```
(venv) occc@occc-VirtualBox:/dj/first_site$ pip install gunicorn psycopg2-binary
Collecting gunicorn
  Using cached gunicorn-20.1.0-py3-none-any.whl (79 kB)
Collecting psycopg2-binary
  Using cached psycopg2_binary-2.9.6-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl
(3.0 MB)
Requirement already satisfied: setuptools>=3.0 in /dj/venv/lib/python3.10/site-packages (from gunicorn) (59.6.0)
Installing collected packages: psycopg2-binary, gunicorn
Successfully installed gunicorn-20.1.0 psycopg2-binary-2.9.6
(venv) occc@occc-VirtualBox:/dj/first_site$
```

Edit the main settings.py to add database connection information

Find the section that configures database access. It will start with **DATABASES**. The configuration in the file is for an SQLite database. We already created a PostgreSQL database for our project, so we need to adjust the settings.

Change the settings with your PostgreSQL database information. We tell Django to use the **psycopg2** adapter we installed with pip. We need to give the database name, username, and user's password and then specify that the database is on the local computer. You can leave the PORT setting as an empty string:

Original DATABASE settings before modification.

```
# Database
# https://docs.djangoproject.com/en/4.2/ref/settings/#databases

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': BASE_DIR / 'db.sqlite3',
    }
}
```

change it to:

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql_psycopg2',
        'NAME': 'orange',
        'USER': 'orangeuser',
        'PASSWORD': 'orange123',
        'HOST': 'localhost',
        'PORT': '',
    }
}

# Database
# https://docs.djangoproject.com/en/4.2/ref/settings/#databases
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql_psycopg2',
        'NAME': 'orange',
        'USER': 'orangeuser',
        'PASSWORD': 'orange123',
        'HOST': 'localhost',
        'PORT': '',
    }
}
#DATABASES = {
#    'default': [
#        {
#            'ENGINE': 'django.db.backends.sqlite3',
#            'NAME': BASE_DIR / 'db.sqlite3',
#        }
#    ]
}
```



4.2 Completing database Setup

we can migrate the initial database schema to our PostgreSQL database using the management script:



occc@occc-sunyorange

```
python manage.py makemigrations
python manage.py migrate
```

```
(orange) occc@occc-VirtualBox:/csc108/project1/django_project/first_lab$ python manage.py makemigrations
No changes detected
(orange) occc@occc-VirtualBox:/csc108/project1/django_project/first_lab$ python manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, sessions
Running migrations:
  Applying contenttypes.0001_initial... OK
  Applying auth.0001_initial... OK
  Applying admin.0001_initial... OK
  Applying admin.0002_logentry_remove_auto_add... OK
  Applying admin.0003_logentry_add_action_flag_choices... OK
  Applying contenttypes.0002_remove_content_type_name... OK
  Applying auth.0002_alter_permission_name_max_length... OK
  Applying auth.0003_alter_user_email_max_length... OK
  Applying auth.0004_alter_user_username_opts... OK
  Applying auth.0005_alter_user_last_login_null... OK
  Applying auth.0006_require_contenttypes_0002... OK
  Applying auth.0007_alter_validators_add_error_messages... OK
  Applying auth.0008_alter_user_username_max_length... OK
  Applying auth.0009_alter_user_last_name_max_length... OK
  Applying auth.0010_alter_group_name_max_length... OK
  Applying auth.0011_update_proxy_permissions... OK
  Applying auth.0012_alter_user_first_name_max_length... OK
  Applying sessions.0001_initial... OK
(orange) occc@occc-VirtualBox:/csc108/project1/django_project/first_lab$ █
```

Create an administrative user for the project by typing:



occc@occc-sunyorange

```
python manage.py createsuperuser
```

```
(orange) occc@occc-VirtualBox:/csc108/project1/django_project/first_lab$ python manage.py createsuperuser
Username (leave blank to use 'occc'): occc
Email address: occc@localhost
Password:
Password (again):
This password is too common.
Bypass password validation and create user anyway? [y/N]: y
Superuser created successfully.
(orange) occc@occc-VirtualBox:/csc108/project1/django_project/first_lab$ █
```

The password was "orange123" and is considered not strong. This password is ok for a lab but not for production.

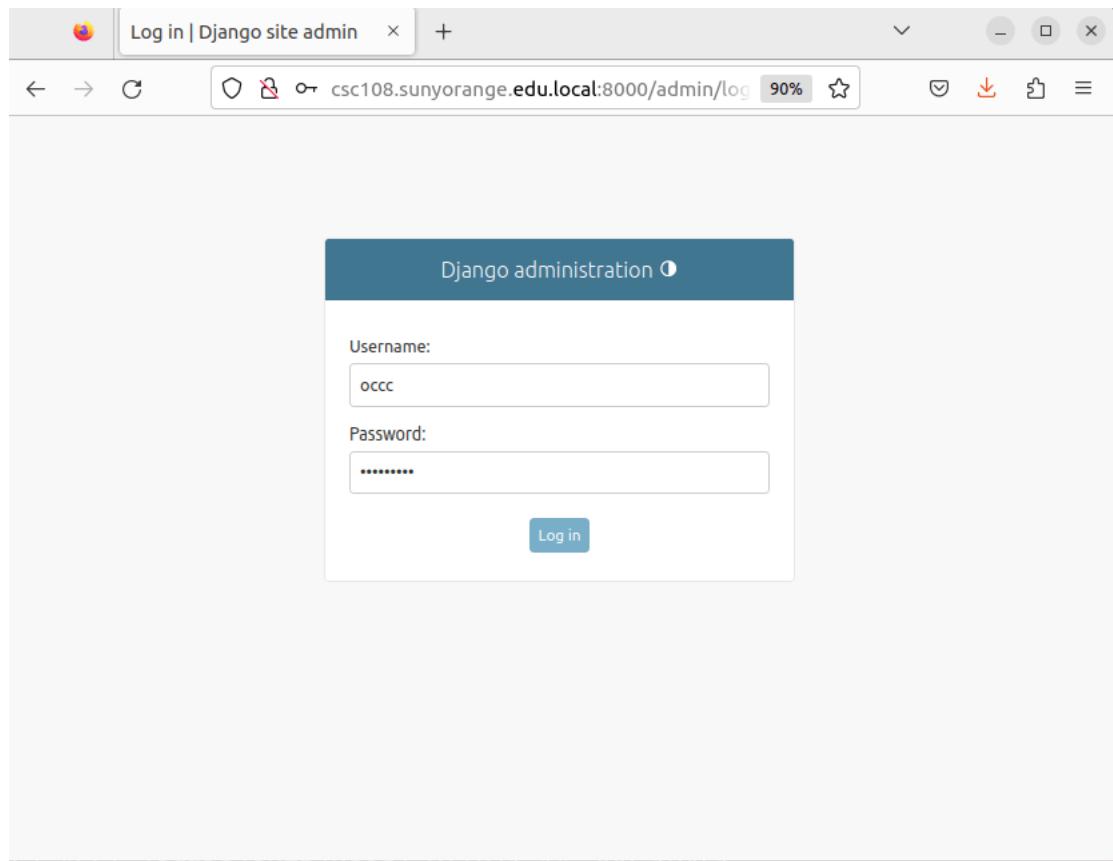
Run your server

```
^C(venv) occc@occc-VirtualBox:/dj/first_site$ python manage.py runserver
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).
July 13, 2023 - 19:12:59
Django version 4.2.3, using settings 'first_site.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CONTROL-C.
```

Notice that we no longer have errors about migrations.

In your browser, navigate to your site "http://127.0.0.1:8000/admin" or, if you set up your allowed hosts, your domain name.



The screenshot shows the Django administration interface for the 'Users' model. The left sidebar under 'AUTHENTICATION AND AUTHORIZATION' has 'Groups' and 'Users' listed. The 'Users' item is selected and highlighted in yellow. The main content area is titled 'Select user to change' and shows a table with one user entry:

	USERNAME	EMAIL ADDRESS	FIRST NAME
<input type="checkbox"/>	occc	occc@localhost	

Below the table, it says '1 user'. On the right, there is a 'FILTER' sidebar with three sections: 'By staff status' (All, Yes, No), 'By superuser status' (All, Yes, No), and 'By active' (All, Yes, No). At the top right of the main content area is a 'ADD USER +' button.



Templates

We had our mainpage app view return a result to the browser. However, this is not a complete HTML page. There were no CSS or header tags present.

The entire page source only contained :

"Hello world from mainpage of csc108 webprogramming 1."

No HTML tags were present. Django works by you creating an HTML template file and use that to serve your content.

5.1

How do templates work in django?

[Django template documentation](#)

In Django, the template can be just a static html file or a html + templating language to create on the fly dynamic websites based on some criterion. Django ships with built-in templating engines for its template system creatively called the **Django template language (DTL)**,) and also the popular alternative **Jinja2** .

create a html template(omit templating for now)

In our mainpage app directory, create a directory called "templates". Then in that templates directory, create another directory with our app name. In this case mainpage. Then inside the template/mainpage directory, create our HTML file. We can name it firsttemplate.html

```

1 !DOCTYPE html>
2 <html lang="en-US">
3   <head>
4     <meta charset="utf-8" >
5     <meta name="description" content="CSC 108 Web programming 1">
6     <meta name="keywords" content="Django, Suny Orange, CSC108">
7     <meta name="author" content="Miroslav Krajca">
8     <meta name="Classification" content='College'>
9     <meta name="copyright" content='Miroslav Krajca'>
10    <title>first template for csc108</title>
11  </head>
12  <body>
13    <h1>Hello World!</h1>
14    <p>Welcome to csc108 mainpage app inside Django!</p>
15  </body>
16 </html>
```



```
(venv) occc@occc-VirtualBox:/dj/first_site/mainpage$ tree
.
├── admin.py
├── apps.py
├── __init__.py
└── migrations
    ├── __init__.py
    └── pycache_
        └── __init__.cpython-310.pyc
├── models.py
└── pycache_
    ├── admin.cpython-310.pyc
    ├── apps.cpython-310.pyc
    ├── __init__.cpython-310.pyc
    ├── models.cpython-310.pyc
    ├── urls.cpython-310.pyc
    └── views.cpython-310.pyc
└── templates
    └── mainpage
        └── firsttemplate.html
├── tests.py
└── urls.py
└── views.py

5 directories, 16 files
(venv) occc@occc-VirtualBox:/dj/first_site/mainpage$
```

edited view.py

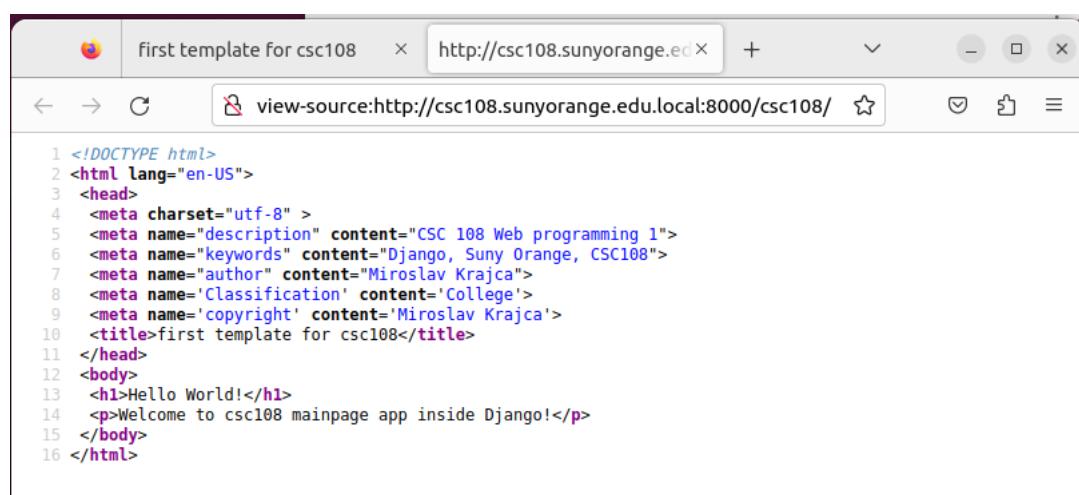
```

1 from django.shortcuts import render
2 from django.http import HttpResponse
3 from django.template import loader
4
5
6 def index(request):
7     template = loader.get_template('mainpage/firsttemplate.html')
8     return HttpResponse(template.render())

```

**Hello World!**

Welcome to csc108 app inside Django!



5.2 ➔ using templates for dynamic websites

Most of the time the content for the web page will be extracted from a database.

passing data to a template

template with variables

```

1 from django.shortcuts import render
2 from django.http import HttpResponseRedirect
3 from django.template import loader
4
5
6 def index(request):
7     """ create a dictionary and pass it to a template """
8     csc108_info = {
9         "name": "Web programming 1",
10        "course_no": "csc108",
11        "location": "Middletown campus",
12    }
13    return render(request, "mainpage/firsttemplate.html", csc108_info)
14 #template = loader.get_template('firsttemplate.html')
15 #return HttpResponseRedirect(template.render())

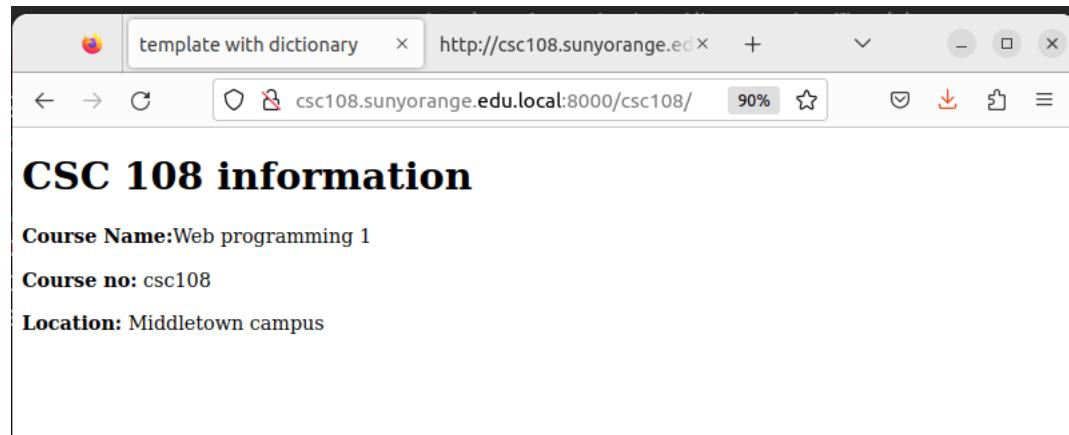
```

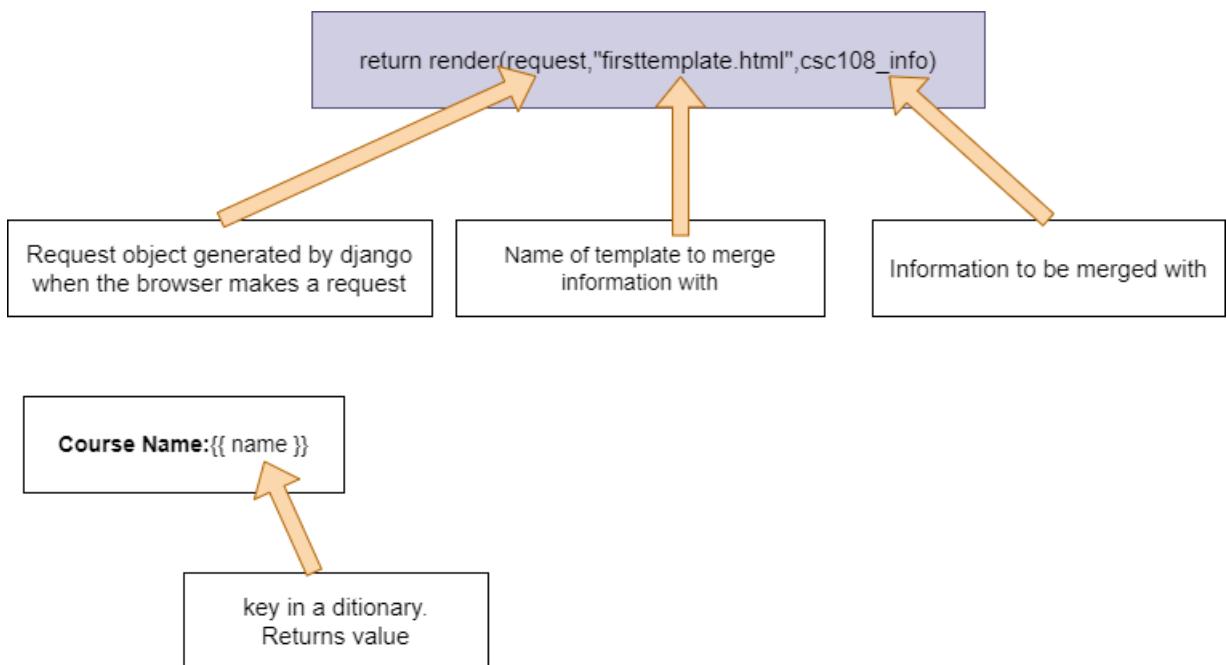
template with variables

```

1 <!DOCTYPE html>
2 <html lang="en-US">
3   <head>
4     <meta charset="utf-8" >
5     <meta name="description" content="CSC 108 Web programming 1">
6     <meta name="keywords" content="Django, Suny Orange, CSC108">
7     <meta name="author" content="Miroslav Krajca">
8     <meta name="Classification" content='College'>
9     <meta name="copyright" content='Miroslav Krajca'>
10    <title>template with dictionary</title>
11  </head>
12  <body>
13    <h1>CSC 108 information</h1>
14    <p><strong> Course Name:</strong>{{ name }}</p>
15    <p><strong> Course no:</strong> {{ course_no }}</p>
16    <p><strong> Location:</strong> {{ location }} </p>
17  </body>
18 </html>

```





5.3 Diff Between {{ }} and % %

- **Interpreted** data, which you note by double braces, {{ value }}
- **Tags**, which you note by braces and percent signs, { % tag_name % }
- **Filters**, which modify interpreted data and you apply with the pipe operator (|), like in {{ value | filter }}

5.3.1 {{ Variables }}

5.3.1.0 dictionaries

In the above code, we saw how to pass a dictionary into a template and then use the values associated with the dictionary to add data to an HTML result.

5.3.1.0 list variables

Lets add two more views to our app.

two new urls in csc108 urls.py

```

1 from django.urls import path
2
3 from . import views
4
5 urlpatterns = [
6     path("", views.index, name="index"),
7     path("dictionary_example/", views.dictionary_example, name="dictionary_example"),
8     path("list_example/", views.list_example, name="list_example"),
9 ]

```

two new views in views.py

```

1 from django.http import HttpResponse
2 from django.template import loader
3
4
5 def index(request):
6     """ create a dictionary and pass it to a template """
7     csc108_info = {
8         "name": "Web programming 1",
9         "course_no": "csc108",
10        "location": "Middletown campus",
11    }
12    return render(request, "mainapage/firsttemplate.html", csc108_info)
13
14
15
16
17 def dictionary_example(request):
18     """ create a dictionary and pass it to a template """
19     context = {
20         "name": "Web programming 1",
21         "course_no": "csc108",
22         "location": "Middletown campus",
23     }
24     return render(request, "mainpage/dictionarytemplate.html", context)
25
26 def list_example(request):
27     """ create a list and display it in a template """
28     mylist = ["Web programming 1", "csc108", "Middletown campus"]
29     return render(request, "mainpage/listtemplate.html", {"items": mylist})

```

listtemplate.html

- An HttpRequest is routed to MyView by the Django URL dispatcher.
- The Django URL dispatcher calls `as_view()` on MyView.
- `as_view()` invokes `setup()` and `dispatch()`.

```

4     dispatch() triggers a method for a specific HTTP method or ↴
      ↴ http_method_not_allowed().
5     An HttpResponse is returned.
6 !DOCTYPE html>
7 <html lang="en-US">
8   <head>
9     <meta charset="utf-8" >
10    <meta name="description" content="CSC 108 Web programming 1">
11    <meta name="keywords" content="Django, Suny Orange, CSC108">
12    <meta name="author" content="Miroslav Krajca">
13    <meta name="Classification" content='College'>
14    <meta name="copyright" content='Miroslav Krajca'>
15    <title>template with dictionary</title>
16  </head>
17  <body>
18    <h1>CSC 108 information</h1>
19    <p><strong> Course Name:</strong>{{ items.0 }}</p>
20    <p><strong> Course no:</strong> {{ items.1 }}</p>
21    <p><strong> Location:</strong> {{ items.2 }} </p>
22  </body>
23 </html>

```

Both the dictionary and the list will produce the same output. As you can see above, we must pass a dictionary into the render method. The key->name corresponds to a list as a value, and the .0 .1 .2 are the indexes in that list.

5.3.2 { % Tags % }

Tags provide arbitrary logic in the rendering process.

This definition is deliberately vague. For example, a tag can output content, serve as a control structure, e.g., an “if” statement or a “for” loop, grab content from a database, or even enable access to other template tags.

Tags are surrounded by { % and % } like this:

{% csrf_token %}

Most tags accept arguments:

{% cycle 'odd' 'even' %}

Some tags require beginning and ending tags:

```
{% if user.is_authenticated %}Hello, {{ user.username }}.{% endif %}
```

Django build in tags

example dictionary template

dictionary template with tags.html

```

1 <!DOCTYPE html>
2 <html lang="en-US">
3   <head>
4     <meta charset="utf-8" >
5     <meta name="description" content="CSC 108 Web programming 1">
6       <meta name="keywords" content="Django, Suny Orange, CSC108">
7     <meta name="author" content="Miroslav Krajca">
8     <meta name="Classification" content='College'>
9     <meta name="copyright" content='Miroslav Krajca'>
10    <title>template with dictionary</title>
11  </head>
12  <body>
13    <h1>CSC 108 information</h1>
14    <!-- classinfo is a key that has a value of a dictionary.-->
15    {% for key, value in classinfo.items %}
16      <p><strong> {{key}}</strong>{{ value }}</p>
17    {% endfor %}
18  </body>
19 </html>
```

PYTHON DICTIONARIES ARE NOT ORDERED!!!!

modified dictionary view

```

1 def dictionary_example(request):
2     """ create a dictionary and pass it to a template """
3     items = {
4         "Course Name": "Web programming 1",
5         "Course no.": "csc108",
6         "Location": "Middletown campus",
7     }
8     context = {"classinfo":items}
9     return render(request,"mainpage/dictionarytemplate.html",context)
```

5.3.3 {{ value | filter }}

Filters transform the values of variables and tag arguments.

Django build in filters



Setup Vscode IDE

6.1 ➤ install vscode

<https://code.visualstudio.com/docs/setup/setup-overview>

On the left-hand side, select your OS.

SETUP

Overview

Linux

macOS

Windows

Raspberry Pi

Network

Additional Components

Enterprise

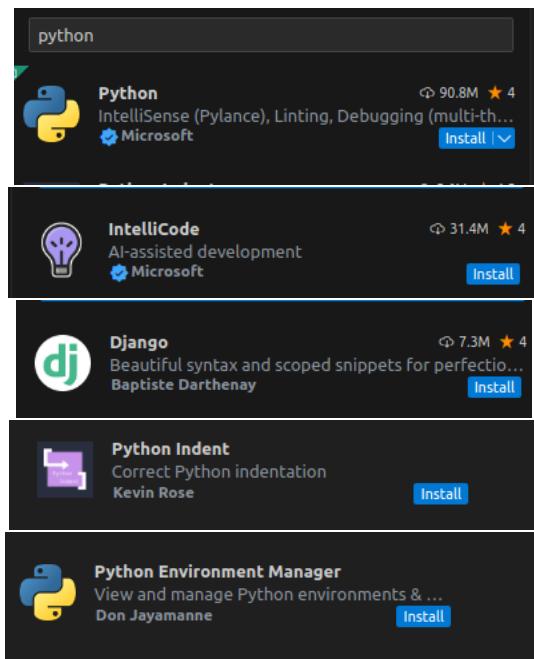
Uninstall

Or go to.

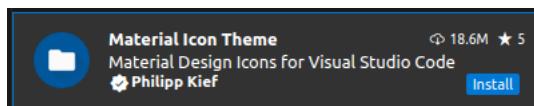
<https://code.visualstudio.com/Download>

6.1.1 install extensions for vscode

Click the extensions button.

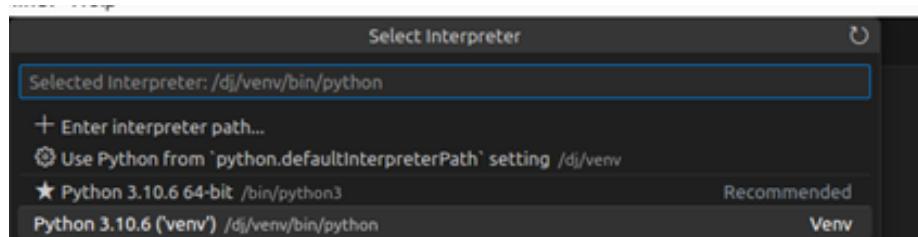


Optional

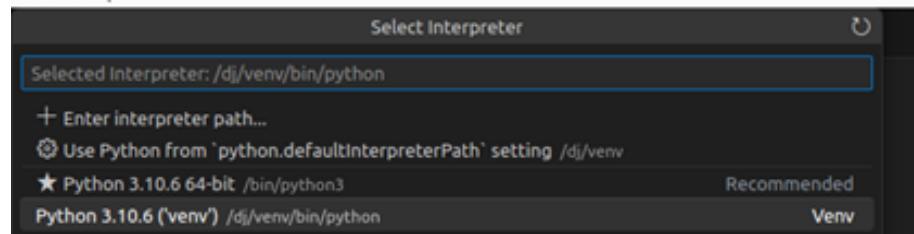


6.2 setup our project

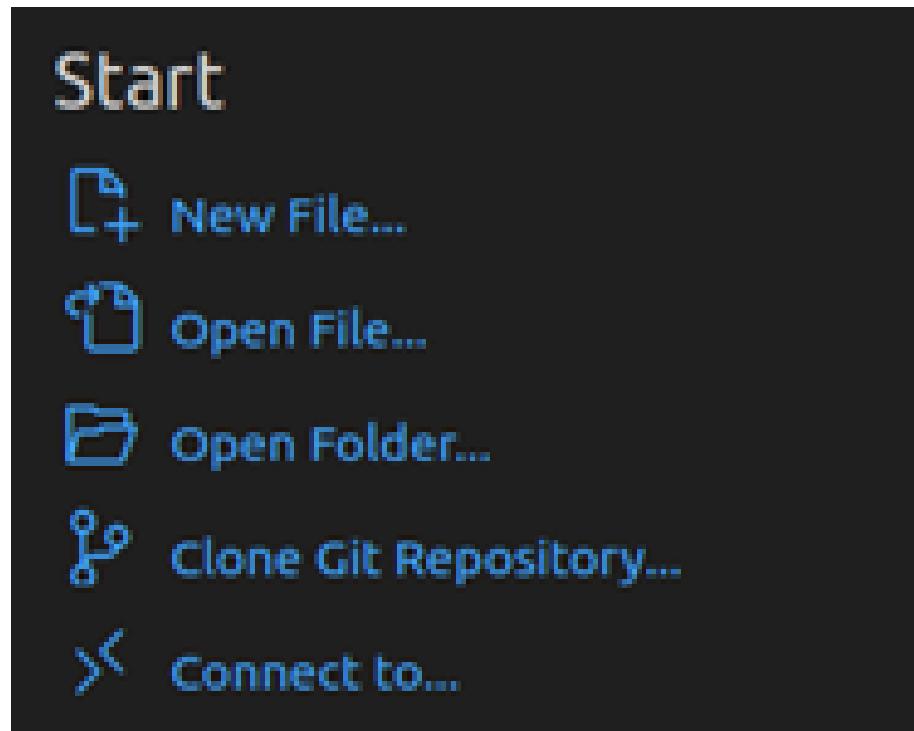
Enter Ctrl+Shift+P in your VScode.



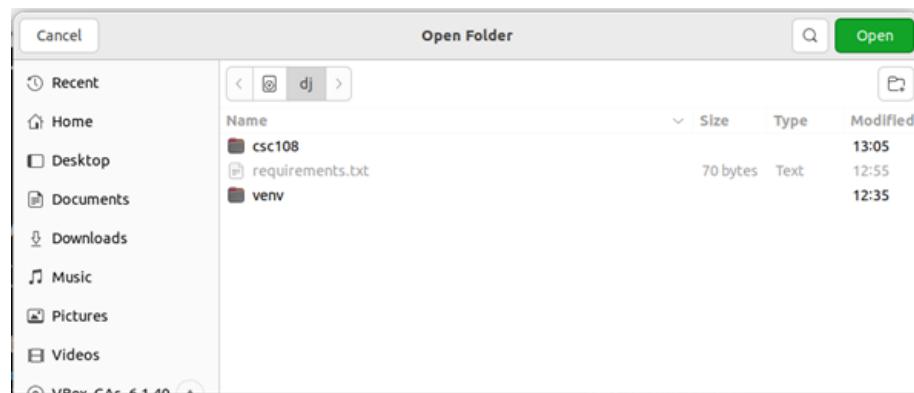
Select Interpreter.

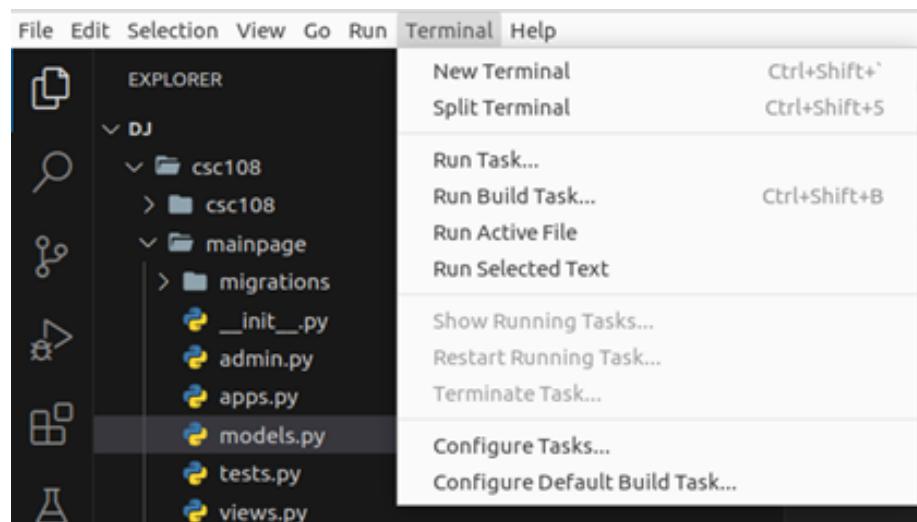
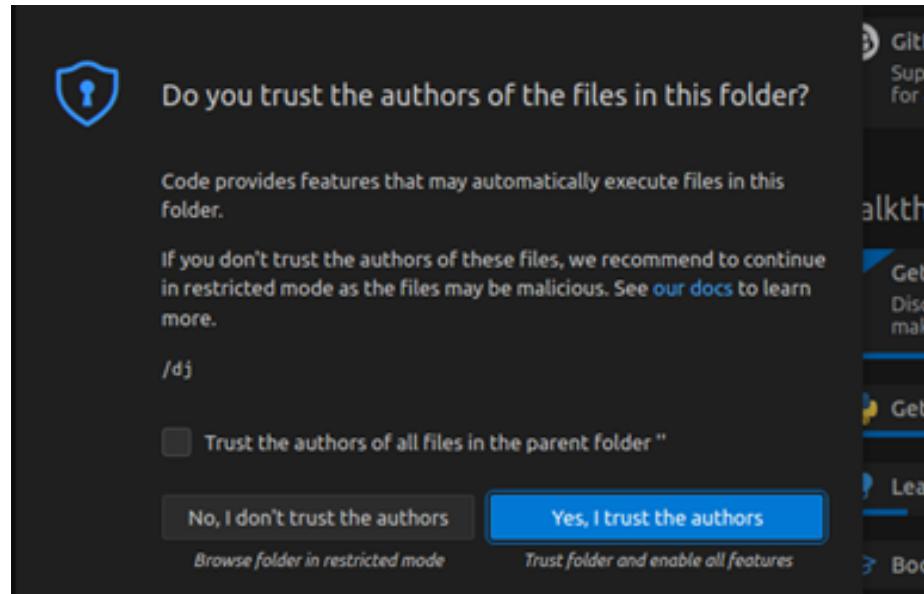


Choose the interpreter that is in your venv. (Helps if the VScode was started after our virtual environment was activated. If not, then navigate to the venv Python location)

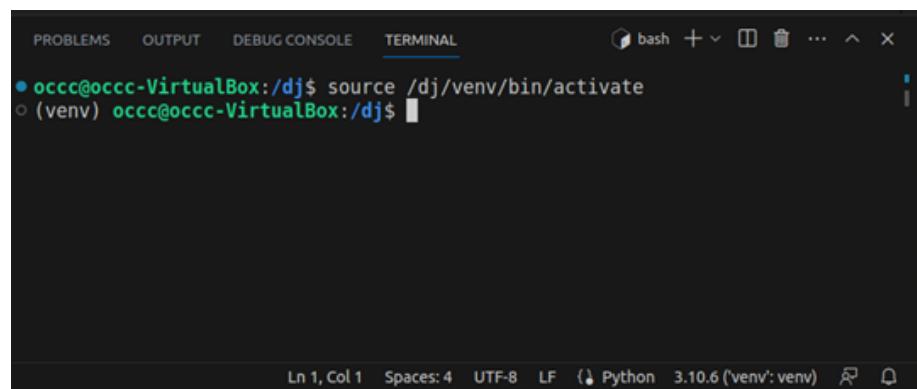


Open our project folder.





Open a new terminal.



Verify that the venv was automatically activated.



college management website

Now that we have some basics on how Django works and how to start creating pages let us try to do an actual website with Django.

7.1 ➤ main landing page

In our mainpage urls.py make sure we have an empty "" view. This empty path is a view without any URL path.

mainpage/urls.py

```
1 from django.urls import path
2
3 from . import views
4
5 urlpatterns = [
6     path("", views.index, name="index"),
7 ]
```

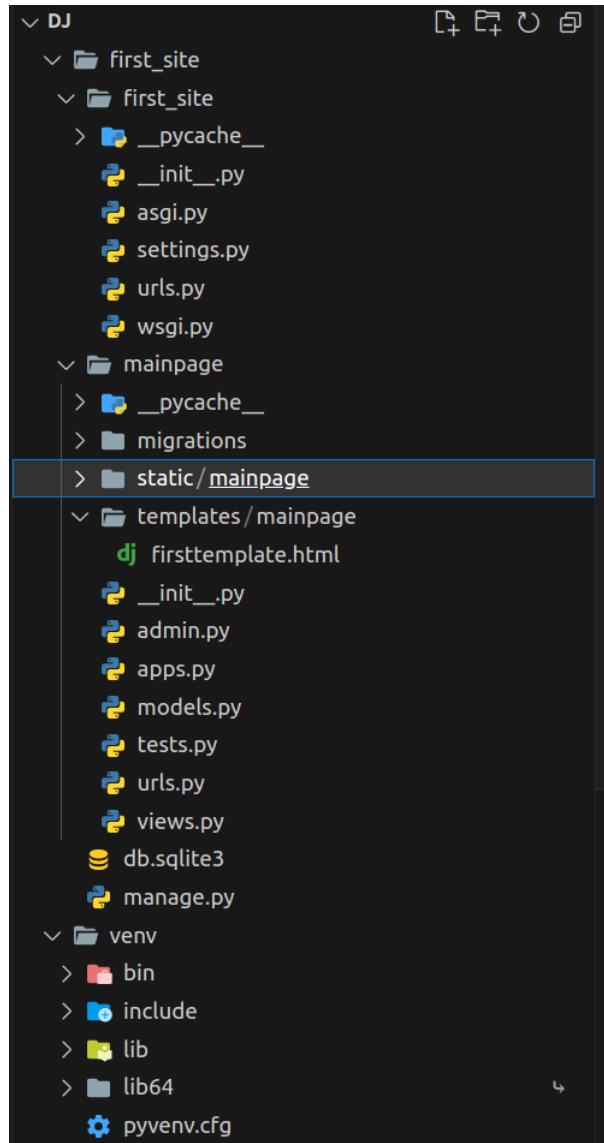
mainpage/views.py

```
1 from django.shortcuts import render
2 from django.http import HttpResponse
3 from django.template import loader
4
5
6 def index(request):
7     """ main landing page for our website """
8     return render(request, "mainpage/mainlandingpage. ↴
         ↴ html")
```

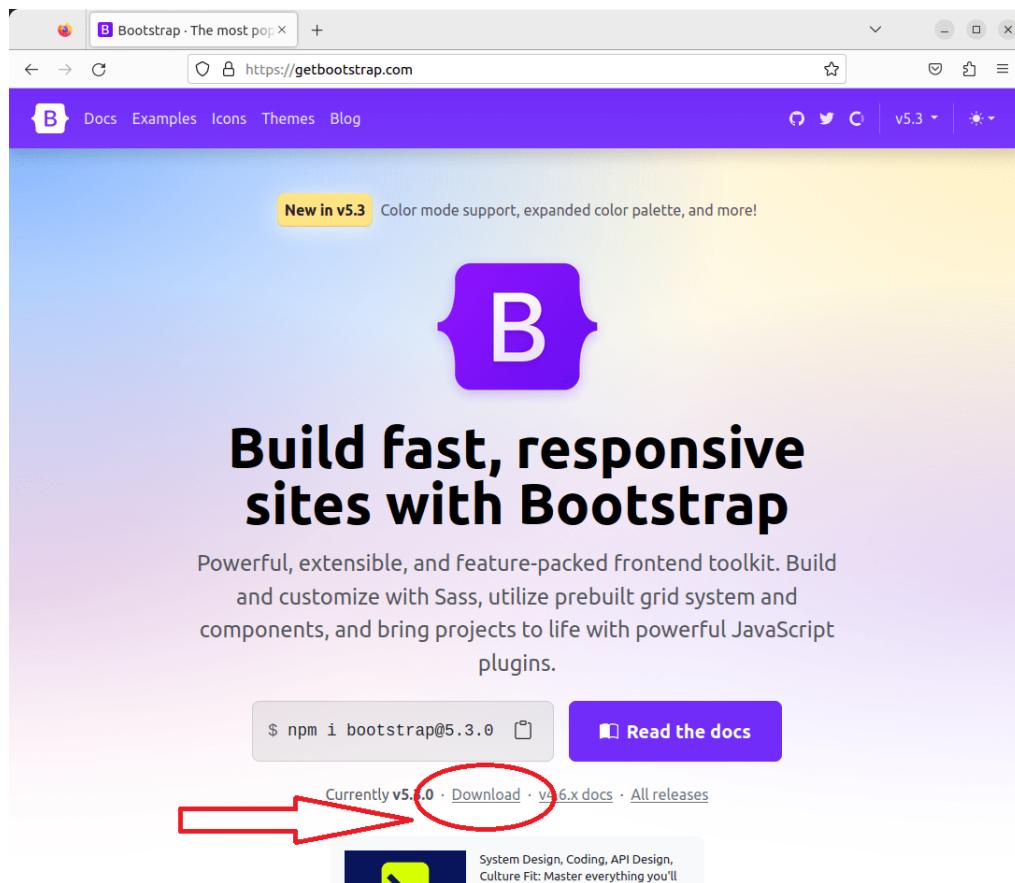
In our app directory, make a static directory.

```
occc@occc-sunyorange:~/Desktop/dj/first_site/mainpage$ mkdir static
occc@occc-sunyorange:~/Desktop/dj/first_site/mainpage$ mkdir static/mainpage
```

We will make a static directory and then make an app name directory as a subdirectory. The above file structure is necessary for later when we merge all the static content into one directory for nginx to serve. Eventually, our mainpage app will look like this regarding the file system.



Download bootstrap CSS and js from <https://getbootstrap.com/>



Compiled CSS and JS

Download ready-to-use compiled code for **Bootstrap v5.3.0** to easily drop into your project, which includes:

- Compiled and minified CSS bundles (see [CSS files comparison](#))
- Compiled and minified JavaScript plugins (see [JS files comparison](#))

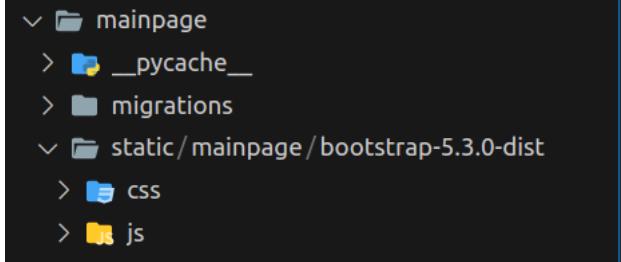
This doesn't include documentation, source files, or any optional JavaScript dependencies like Popper.

[Download](#)

```
occc@occc-VirtualBox:~/Downloads$ ls
bootstrap-5.3.0-dist.zip
occc@occc-VirtualBox:~/Downloads$
```

Extract the contents into our static directory.

```
(venv) occc@occc-sunyorange:~/mainpage$ unzip /home/occc/Downloads/bootstrap-5.3.0-dist.zip
Archive: /home/occc/Downloads/bootstrap-5.3.0-dist.zip
  creating: bootstrap-5.3.0-dist/
  creating: bootstrap-5.3.0-dist/css/
  inflating: bootstrap-5.3.0-dist/css/bootstrap-grid.css
  inflating: bootstrap-5.3.0-dist/css/bootstrap-grid.css.map
...
  inflating: bootstrap-5.3.0-dist/js/bootstrap.min.js.map
(venv) occc@occc-VirtualBox:/dj/first_site/mainpage/static/mainpage$
```



```

    <-- mainpage
      > __pycache__
      > migrations
      <-- static/mainpage/bootstrap-5.3.0-dist
          > css
          > js
  
```

Configure our static files for our Django project

<https://docs.djangoproject.com/en/4.2/howto/static-files/>

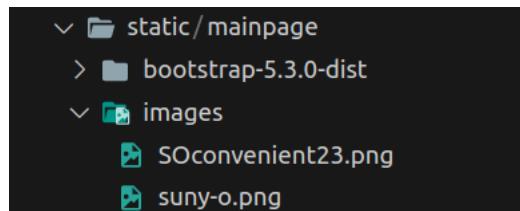
Make sure that `STATIC_URL` is defined in our settings file.

```
# Static files (CSS, JavaScript, Images)
# https://docs.djangoproject.com/en/4.2/howto/static-files/

STATIC_URL = 'static/'
```

Download a logo we can use in our webpage that might suit a navigation bar. Create a sub-directory in our static directory called images and place our logo there.

```
(venv) occc@occc-VirtualBox:/dj/first_site/mainpage/static/mainpage$ mkdir images
(venv) occc@occc-VirtualBox:/dj/first_site/mainpage/static/mainpage$ cd images
(venv) ...:/dj/first_site/mainpage/static/mainpage/images$ cp ~/Downloads/suny-o.png .
(venv) ...:/dj/first_site/mainpage/static/mainpage/images$ cp ~/Downloads/SOconvenient23.png .
(venv) occc@occc-VirtualBox:/dj/first_site/mainpage/static/mainpage/images$ ls
SOconvenient23.png  suny-o.png
(venv) occc@occc-VirtualBox:/dj/first_site/mainpage/static/mainpage/images$
```



We can get several more images or create our own.
Create our website.

```
mainlandingpage.html

1  {% load static %} 
2  <!DOCTYPE html>
3  <html lang="en-US">
4      <head>
5          <meta charset="utf-8">
6          <meta name="description" content="CSC 108 Web programming 1">
7          <meta name="keywords" content="Django, Suny Orange, CSC108">
8          <meta name="author" content="Miroslav Krajca">
9          <meta name="Classification" content='College'>
10         <meta name="copyright" content='Miroslav Krajca'>
11         <title>College main web page</title>
12         <link rel="stylesheet"
13             type="text/css"
14             href="{% static 'mainpage/bootstrap-5.3.0-dist/css/bootstrap.min.css' %}">
15     </head>
16     <body>
17         <nav class="navbar fixed-top navbar-light"
18             style="background-color: #e3f2fd">
19             <a class="navbar-brand" href="#">
20                 
25                 Suny Orange
26             </a>
27         </nav>
28         
31         <footer class="text-center text-lg-start text-white fixed-bottom"
32             style="background-color: #1c2331">
33             <!-- Section: Links -->
34             <section class="">
35                 <div class="container text-center text-md-start mt-5">
36                     <!-- Grid row -->
37                     <div class="row mt-3">
38                         <!-- Grid column -->
39                         <div class="col-md-3 col-lg-4 col-xl-3 mx-auto mb-4">
40                             <!-- Content -->
41                             <h6 class="text-uppercase fw-bold">Suny Orange</h6>
42                             <hr class="mb-4 mt-0 d-inline-block mx-auto"
43                                 style="width: 60px;
44                                     background-color: #7c4dff;
45                                     height: 2px" />
46                             <p>
47                             <strong>Middletown campus:</strong>
48                         </p>
49                         <p>
50                             <strong>Newburgh campus:</strong>
```

```
51      </p>
52  </div>
53  <!-- Grid column -->
54  <div class="col-md-2 col-lg-2 col-xl-2 mx-auto mb-4">
55    <!-- Links -->
56    <h6 class="text-uppercase fw-bold">Academics</h6>
57    <hr class="mb-4 mt-0 d-inline-block mx-auto"
58      style="width: 60px;
59        background-color: #7c4dff;
60        height: 2px" />
61    <p>
62      <a href="#" class="text-white">Registrar</a>
63    </p>
64    <p>
65      <a href="https://catalog.sunyorange.edu/current/programs/ ↴
66        index.html"
67          class="text-white">Degrees</a>
68    </p>
69    <p>
70      <a href="https://sunyorange.edu/financialaid/index.html"
71          class="text-white">Financial Aid</a>
72    </p>
73  </div>
74  <!-- Grid column -->
75  <!-- Grid column -->
76  <div class="col-md-3 col-lg-2 col-xl-2 mx-auto mb-4">
77    <!-- Links -->
78    <h6 class="text-uppercase fw-bold">Useful links</h6>
79    <hr class="mb-4 mt-0 d-inline-block mx-auto"
80      style="width: 60px;
81        background-color: #7c4dff;
82        height: 2px" />
83    <p>
84      <a href="#" class="text-white">Your Account</a>
85    </p>
86    <p>
87      <a href="#" class="text-white">News</a>
88    </p>
89    <p>
90      <a href="#" class="text-white">Help</a>
91    </p>
92  </div>
93  <!-- Grid column -->
94  <!-- Grid column -->
95  <div class="col-md-4 col-lg-3 col-xl-3 mx-auto mb-md-0 mb-4">
96    <!-- Links -->
97    <h6 class="text-uppercase fw-bold">Contact</h6>
98    <hr class="mb-4 mt-0 d-inline-block mx-auto"
99      style="width: 60px;
100        background-color: #7c4dff;
101        height: 2px" />
102    <p>Middletown Campus</p>
103    <p>115 South Street</p>
104    <p>Middletown, NY 10940</p>
105  </div>
106  <!-- Grid column -->
107</div>
```

```

107      <!-- Grid row -->
108      </div>
109    </section>
110    <!-- Section: Links -->
111    <!-- Copyright -->
112    <div class="text-center p-3" style="background-color: rgba(0, 0, 0, 0.2)">
113      © 2023 Copyright:
114      <a class="text-white" href="http://csc108.sunyorange.edu.local/">
115        CSC 108 Web Programming 1</a>
116    </div>
117    <!-- Copyright -->
118  </footer>
119  <!-- Footer -->
120 </body>
121 </html>

```

The screenshot shows a web browser window with the URL `csc108.sunyorange.edu.local:8000/csc108/#`. The page has a blue header with the Suny Orange logo. The main content area features a large orange button with the text '#SOconvenient' and a subtext 'Our campuses are conveniently located in Middletown and Newburgh.' Below this is a photograph of a modern building with glass windows and a red brick base, labeled 'KAPLAN HALL'. The footer is dark with white text, organized into four sections: SUNY ORANGE (links to Middletown and Newburgh campuses), ACADEMICS (links to Registrar, Degrees, and Financial Aid), USEFUL LINKS (links to Your Account, News, and Help), and CONTACT (information about the Middletown Campus at 115 South Street, Middletown, NY 10940). At the bottom of the footer, it says '© 2023 Copyright: CSC 108 Web Programming 1'.

7.2 ➤ static

```
{% load static %}
```

In Django, `{% load static %}` is a template tag used to initialize the static template tags, which allow your templates to access static files.

Static files, in this context, are not dynamically generated files that already exist on your server, such as CSS files, JavaScript files, images, or any other types of files that your web page might need to reference. These files are typically not changed by the server after they are created, and they are the same for every user that requests them, unlike dynamic content, which may be different for each user or each request.

This tag allows you to embed links for static files.

static tag

```

```

From our log file, we can see this is translated to:

```
[14/Jul/2023 14:55:14] "GET /static/mainpage/images/suny-o.png HTTP/1.1" 304 0
```

We can consider this our public-facing website. Next, we will add our secure login website with more information and dynamically display data based on who is logged in.

When we move this to Nginx, we will have to do some minor things to serve static content from Nginx.

7.3 ➤ split up page for better control

The above code does what we need it to do; however, it will be difficult to manage later. With a long HTML page, file editing and adding items will be progressively more and more difficult. We can split this up into several files.

7.3.1 ➤ base html file

`mainpage/urls.py`

```

1 from django.urls import path
2
3 from . import views
4
5 urlpatterns = [
6     path("index", views.index, name="index"),
7     path("",views.index, name="home"),
8 ]

```

`views.py`

```

1 from django.shortcuts import render
2 from django.http import HttpResponseRedirect
3 from django.template import loader
4
5
6 def index(request):
7     """ main landing page for our website """
8     return render(request, "mainpage/mainlandingpage.2"
9                   ↴ html")
10
11 def home(request):
12     return render(request, "mainpage/home.html")

```

`mainpage/templates/mainpage/base.html`

```

1 {% load static %}
2 <!DOCTYPE html>
3 <html lang="en-US">
4     <head>
5         <meta charset="utf-8">
6         <meta name="description" content="CSC 108 Web programming 1">
7         <meta name="keywords" content="Django, Suny Orange, CSC108">
8         <meta name="author" content="Miroslav Krajca">
9         <meta name="Classification" content='College'>
10        <meta name="copyright" content='Miroslav Krajca'>
11        <title>{% block title %}{% endblock title %}</title>
12        <link rel="stylesheet"
13              type="text/css"

```

```

14         href="{% static 'mainpage/bootstrap-5.3.0-dist/css/bootstrap.min.css' %}">
15     {% block custom_css %}{% endblock custom_css %}
16 </head>
17 <body>
18     {% include 'mainpage/includes/navbar.html' %}
19     <main role="main" class="container-fluid">
20         <div class="row">
21             <div class="col-lg-12 col-md-12 p-0 col-sm-12">
22                 {% include 'mainpage/includes/messaging.html' %}
23                 {% block content %}{% endblock content %}
24             </div>
25         </div>
26     </main>
27     <!-- Optional Javascript -->
28     <script src="{% static 'mainpage/jquery3.7.0/jquery-3.7.0.min.js' %}"></script>
29     <script src="{% static 'mainpage/bootstrap-5.3.0-dist/js/bootstrap.min.js' %}"></script>
30     {% include 'mainpage/includes/footer.html' %}
31     <script>
32         $(document).ready(function() {
33             window.setTimeout(function() {
34                 $(".alert").fadeTo(500, 0).slideUp(500, function() {
35                     $(this).remove();
36                 });
37             }, 5000);
38         });
39     </script>
40     {% block custom_js %} {% endblock custom_js %}
41 </body>
42 </html>

```

The **block** tag defines blocks that child templates can fill in. All the block tag does is tell the template engine that a child template may override those portions.

include Loads a template and renders it with the current context. The include tag is a way of “including” other templates within a template and merging them.

The template name can either be a variable or a hard-coded (quoted) string in single or double quotes.

mainpage/templates/mainpage/home.html

```

1  {% extends "mainpage/base.html" %} 
2  {% load static %} 
3  {% block title %} 
4  College Main web page 
5  {% endblock title %} 
6  {% block custom_css %} 
7  {% endblock custom_css %} 
8 
9  {% block content %} 
10  
11 {% endblock content %} 
12 
13 {% block custom_js %} {% endblock custom_js %}
```

The extends tag tells the template engine that this template “extends” another template. When the template system evaluates this template, first, it locates the parent – in this case, “base.html”.

mainpage/templates/mainpage/includes/navbar.html

```

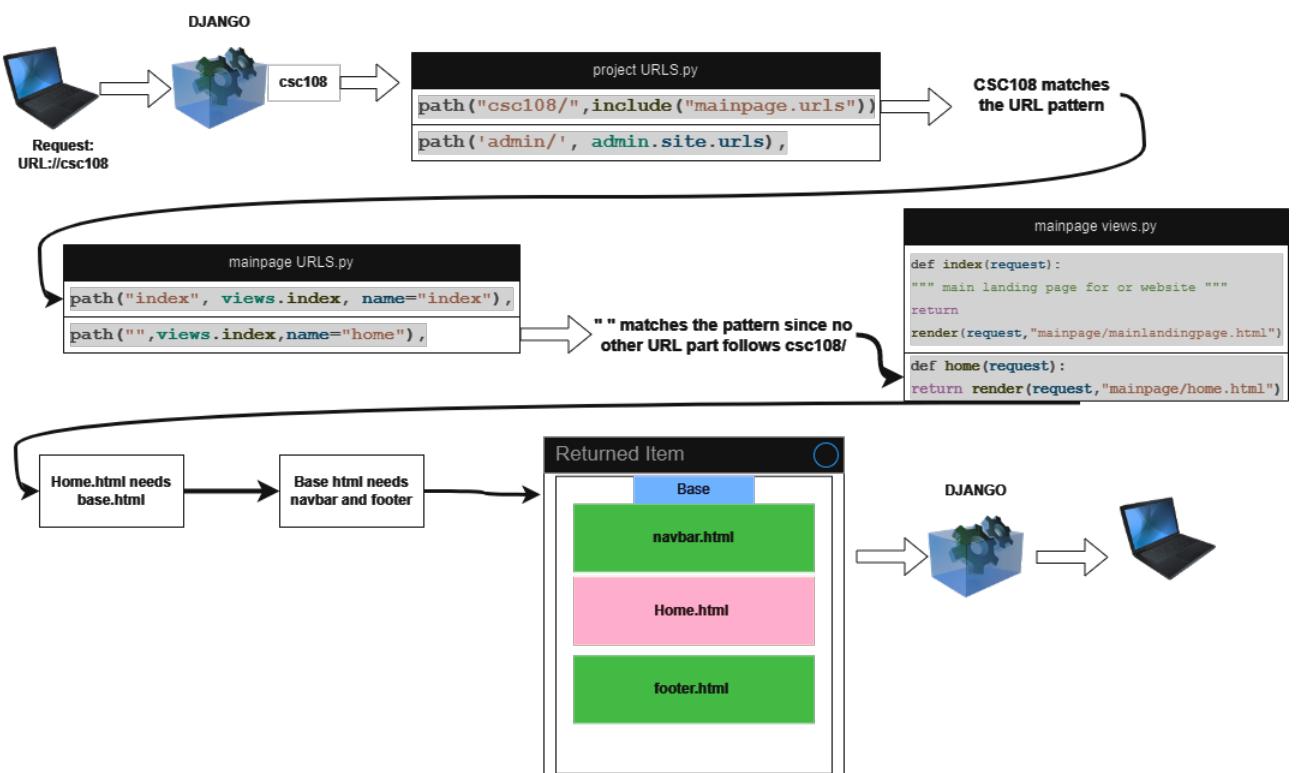
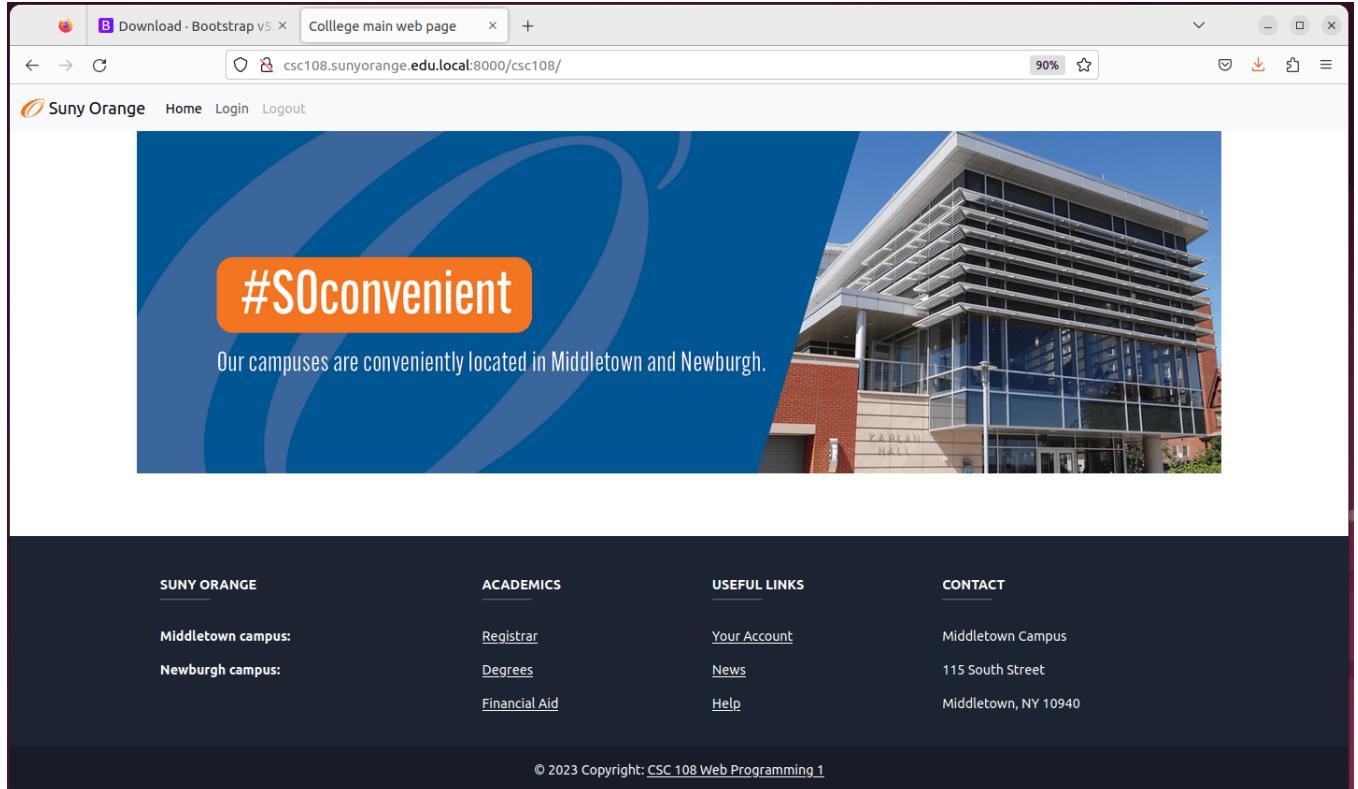
1  {% load static %} 
2  <nav class="navbar navbar-expand-lg bg-body-tertiary" 
3   style="background-color: #e3f2fd"> 
4   <div class="container-fluid"> 
5     <a class="navbar-brand" href="#"> 
6        
11       Sunny Orange 
12     </a> 
13     <button class="navbar-toggler" 
14       type="button" 
15       data-bs-toggle="collapse" 
16       data-bs-target="#navbarTogglerDemo03" 
17       aria-controls="navbarTogglerDemo03" 
18       aria-expanded="false" 
19       aria-label="Toggle navigation"> 
20       <span class="navbar-toggler-icon"></span> 
21     </button> 
22     <div class="collapse navbar-collapse" id="navbarTogglerDemo03"> 
23       <ul class="navbar-nav me-auto mb-2 mb-lg-0"> 
24         <li class="nav-item"> 
25           <a class="nav-link active" aria-current="page" href="#">Home</a> 
26         </li> 
27         <li class="nav-item"> 
28           <a class="nav-link" href="#">Login</a> 
29         </li> 
30         <li class="nav-item"> 
31           <a class="nav-link disabled">Logout</a>
```

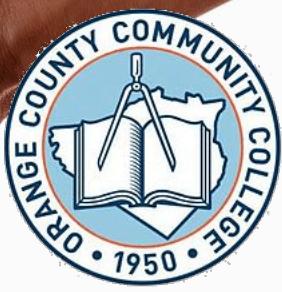
```
32      </li>
33    </ul>
34  </div>
35 </div>
36 </nav>
```

mainpage/templates/mainpage/includes/footer.html

```
1  {% load static %}
2  <footer class="text-center text-lg-start text-white fixed-bottom"
3      style="background-color: #1c2331">
4    <!-- Section: Links -->
5    <section class="">
6      <div class="container text-center text-md-start mt-5">
7        <!-- Grid row -->
8        <div class="row mt-3">
9          <!-- Grid column -->
10         <div class="col-md-3 col-lg-4 col-xl-3 mx-auto mb-4">
11           <!-- Content -->
12           <h6 class="text-uppercase fw-bold">Suny Orange</h6>
13           <hr class="mb-4 mt-0 d-inline-block mx-auto"
14             style="width: 60px;
15                 background-color: #7c4dff;
16                 height: 2px" />
17           <p>
18             <strong>Middletown campus:</strong>
19           </p>
20           <p>
21             <strong>Newburgh campus:</strong>
22           </p>
23         </div>
24         <!-- Grid column -->
25         <!-- Grid column -->
26         <div class="col-md-2 col-lg-2 col-xl-2 mx-auto mb-4">
27           <!-- Links -->
28           <h6 class="text-uppercase fw-bold">Academics</h6>
29           <hr class="mb-4 mt-0 d-inline-block mx-auto"
30             style="width: 60px;
31                 background-color: #7c4dff;
32                 height: 2px" />
33           <p>
34             <a href="#" class="text-white">Registrar</a>
35           </p>
36           <p>
37             <a href="https://catalog.sunyorange.edu/current/programs/index.html"
38               class="text-white">Degrees</a>
39           </p>
40           <p>
41             <a href="https://sunyorange.edu/financialaid/index.html"
42               class="text-white">Financial Aid</a>
43           </p>
44         </div>
45         <!-- Grid column -->
46         <!-- Grid column -->
```

```
47      <div class="col-md-3 col-lg-2 col-xl-2 mx-auto mb-4">
48          <!-- Links -->
49          <h6 class="text-uppercase fw-bold">Useful links</h6>
50          <hr class="mb-4 mt-0 d-inline-block mx-auto"
51              style="width: 60px;
52                  background-color: #7c4dff;
53                  height: 2px" />
54          <p>
55              <a href="#" class="text-white">Your Account</a>
56          </p>
57          <p>
58              <a href="#" class="text-white">News</a>
59          </p>
60          <p>
61              <a href="#" class="text-white">Help</a>
62          </p>
63      </div>
64      <!-- Grid column -->
65      <!-- Grid column -->
66      <div class="col-md-4 col-lg-3 col-xl-3 mx-auto mb-md-0 mb-4">
67          <!-- Links -->
68          <h6 class="text-uppercase fw-bold">Contact</h6>
69          <hr class="mb-4 mt-0 d-inline-block mx-auto"
70              style="width: 60px;
71                  background-color: #7c4dff;
72                  height: 2px" />
73          <p>Middletown Campus</p>
74          <p>115 South Street</p>
75          <p>Middletown, NY 10940</p>
76      </div>
77      <!-- Grid column -->
78  </div>
79  <!-- Grid row -->
80 </div>
81 </section>
82 <!-- Section: Links -->
83 <!-- Copyright -->
84 <div class="text-center p-3" style="background-color: rgba(0, 0, 0, 0.2)">
85     <!-- Copyright: -->
86     <a class="text-white" href="http://csc108.sunyorange.edu.local/">CSC
87         <!-- 108 Web Programming 1-->
88     </a>
89 </div>
90 <!-- Copyright -->
91 </footer>
92 <!-- Footer -->
```





Authentication

8.1 ➤

What is user authentication?

User authentication is the process of verifying the identity of a user or entity attempting to access a system, device, or application. It is a crucial security measure implemented to ensure that only authorized individuals or entities are granted access to protected resources or information.

Authentication typically involves the following steps:

- **User identification:** The user provides a unique identifier, such as a username or email address, to indicate their identity.
- **Credential submission:** The user submits a credential, usually a password, which is associated with their account or identity.
- **Verification:** The system compares the provided credential with the stored credentials associated with the user's identity. This is typically done by comparing the submitted password with the previously encrypted or hashed version stored in the system's database.
- **Authentication decision:** Based on the comparison result, the system determines whether the user is authenticated or not. If the provided credential matches the stored one, the user is granted access. Otherwise, access is denied.

Authentication can also involve additional factors beyond just a password, such as:

types of Authentication

- **Two-factor authentication (2FA):** This involves using a second verification method, in addition to a password, such as a one-time password (OTP) sent to a registered mobile device or a biometric factor like a fingerprint or facial recognition.
- **Multi-factor authentication (MFA):** Similar to 2FA, MFA adds additional layers of authentication, such as a combination of something the user knows (e.g., password), something the user has (e.g., a physical security key), and something the user is (e.g., biometric factors).
- **Token-based authentication:** In this method, a unique token is generated and issued to the user upon successful authentication. The token is then used for subsequent requests, eliminating the need to provide credentials repeatedly.

User authentication is critical for protecting sensitive data, preventing unauthorized access, and ensuring the security and privacy of users' accounts and information. It is widely used in various systems and applications, including websites, online services, mobile devices, and enterprise networks.

8.2

Web Authentication Methods Compared

8.2.1

Authentication vs Authorization

Authentication and authorization are two distinct but closely related concepts in the realm of computer security. While authentication verifies the identity of a user or entity, authorization determines what actions or resources the authenticated user is allowed to access.

Authentication:

Authentication is the process of verifying the identity of a user or entity. It ensures that the individual or entity claiming a particular identity is, in fact, who they say they are. Authentication is typically based on credentials provided by the user, such as a username and password combination. The goal is to validate that the user's claimed identity matches the identity stored in the system.

The primary focus of authentication is answering the question, "Who are you?" It confirms the user's identity to establish trust and enable access to a system or application. Successful authentication results in the user being granted access to the system, while failed authentication leads to denial of access.

Authorization:

Authorization, on the other hand, is the process of determining what actions or resources an authenticated user is permitted to access. Once a user's identity has been authenticated, authorization defines the privileges and permissions associated with that identity. It establishes the boundaries of access for the authenticated user based on their role, level of trust, or other predefined criteria.

The key question authorization addresses is, "What are you allowed to do?" It involves evaluating the user's permissions and access rights to ensure they can perform specific actions or access certain resources within the system. Authoriza-

tion controls are typically based on policies, rules, or permissions defined by the system administrator or application developer.

In summary, authentication is the process of verifying identity, while authorization determines the actions or resources that an authenticated user can access. Authentication establishes trust and verifies the user's identity, while authorization defines the boundaries of what the authenticated user is allowed to do within the system. Both processes work together to ensure secure and controlled access to computer systems and resources.

8.2.2 HTTP Basic Authentication

Basic authentication, which is built into the HTTP protocol, is the most basic form of authentication. With it, login credentials are sent in the request headers with each request:

”Authorization: Basic b2NjYzpvcmFuZ2U=” csc108.sunyorange.edu.local

Usernames and passwords are not encrypted. Instead, the username and password are concatenated together using a **:** symbol to form a single string: **username:password**. This string is then encoded using base64.

base64 encode/decode

```

1 import base64
2 auth = "occc:orange"
3 auth_bytes = auth.encode('ascii') # convert to bytes
4 print(auth_bytes)
5 encoded = base64.b64encode(auth_bytes) # base64 encode
6 print(encoded)
7 print(base64.b64decode(encoded)) # base64 decode

```

Output

```

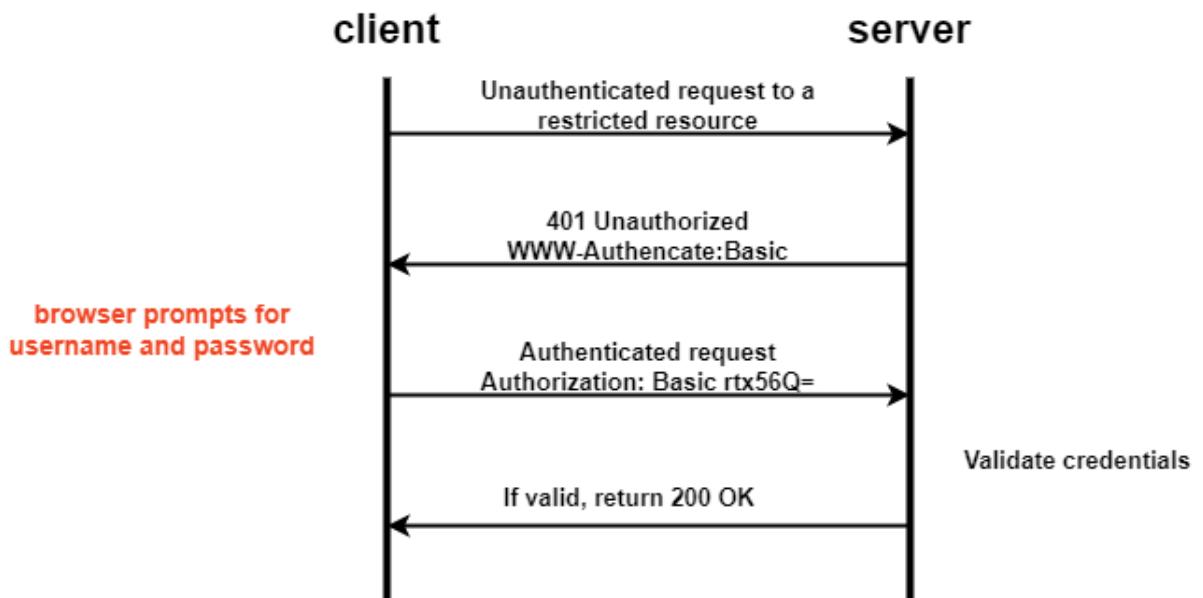
b'occc:orange'
b'b2NjYzpvcmFuZ2U='
b'occc:orange'

```

This method is stateless, so the client must supply the credentials with each and every request. It's suitable for API calls along with simple auth workflows that do not require persistent sessions.

Flow

- Unauthenticated client requests a restricted resource
- HTTP 401 Unauthorized is returned with a header WWW-Authenticate that has a value of Basic.
- The WWW-Authenticate: Basic header causes the browser to display the username and password prompt
- After entering your credentials, they are sent in the header with each request: Authorization: Basic b2NjYzpvcmFuZ2U=



Pros

- Since there aren't many operations going on, authentication can be faster with this method.
- Easy to implement.
- Supported by all major browsers.

Cons

- Base64 is not the same as encryption. It's just another way to represent data. The base64 encoded string can easily be decoded since it's sent in plain text. This poor security feature calls for many types of attacks. Because of this, HTTPS/SSL is absolutely essential.
- Credentials must be sent with every request.
- Users can only be logged out by rewriting the credentials with an invalid one.

Packages

- <https://studygyaan.com/django/basic-authentication-in-django>
- <https://fastapi.tiangolo.com/advanced/security/http-basic-auth/>

8.2.3 HTTP Digest Authentication

HTTP Digest Authentication is a method used for user authentication in HTTP (Hypertext Transfer Protocol) communication. It is designed to provide a more secure alternative to Basic Authentication, which transmits usernames and passwords in plain text.

In HTTP Digest Authentication, the client sends a request to access a protected resource on a server. The server responds with a challenge containing a unique string, known as a nonce (number used once). The client then calculates a hash (digest) of the username, password, and other request information using a cryptographic algorithm, such as MD5 or SHA-256, along with the nonce.

The client sends the calculated hash, along with the username and other relevant information, in another request to the server. This second request is known as the authentication request. The server also performs the same hash calculation using the stored password and compares it to the hash received from the client.

If the calculated hashes match, the server considers the user authenticated and

grants access to the requested resource. However, if the hashes don't match or the authentication request is not properly constructed, the server denies access.

HTTP Digest Authentication offers several security advantages:

- **Protection against eavesdropping:** Since the password is not transmitted in plain text, it is resistant to interception by attackers sniffing network traffic.
- **Non-replayability:** The server includes a nonce in the challenge, which helps prevent replay attacks where an attacker captures and reuses an authentication request.
- **Authentication without storing plaintext passwords:** The server does not need to store passwords in plaintext form. Instead, it stores a hash or digest of the password, enhancing security in case of a data breach.
- **Server authentication:** HTTP Digest Authentication can also verify the server's identity, protecting against man-in-the-middle attacks.

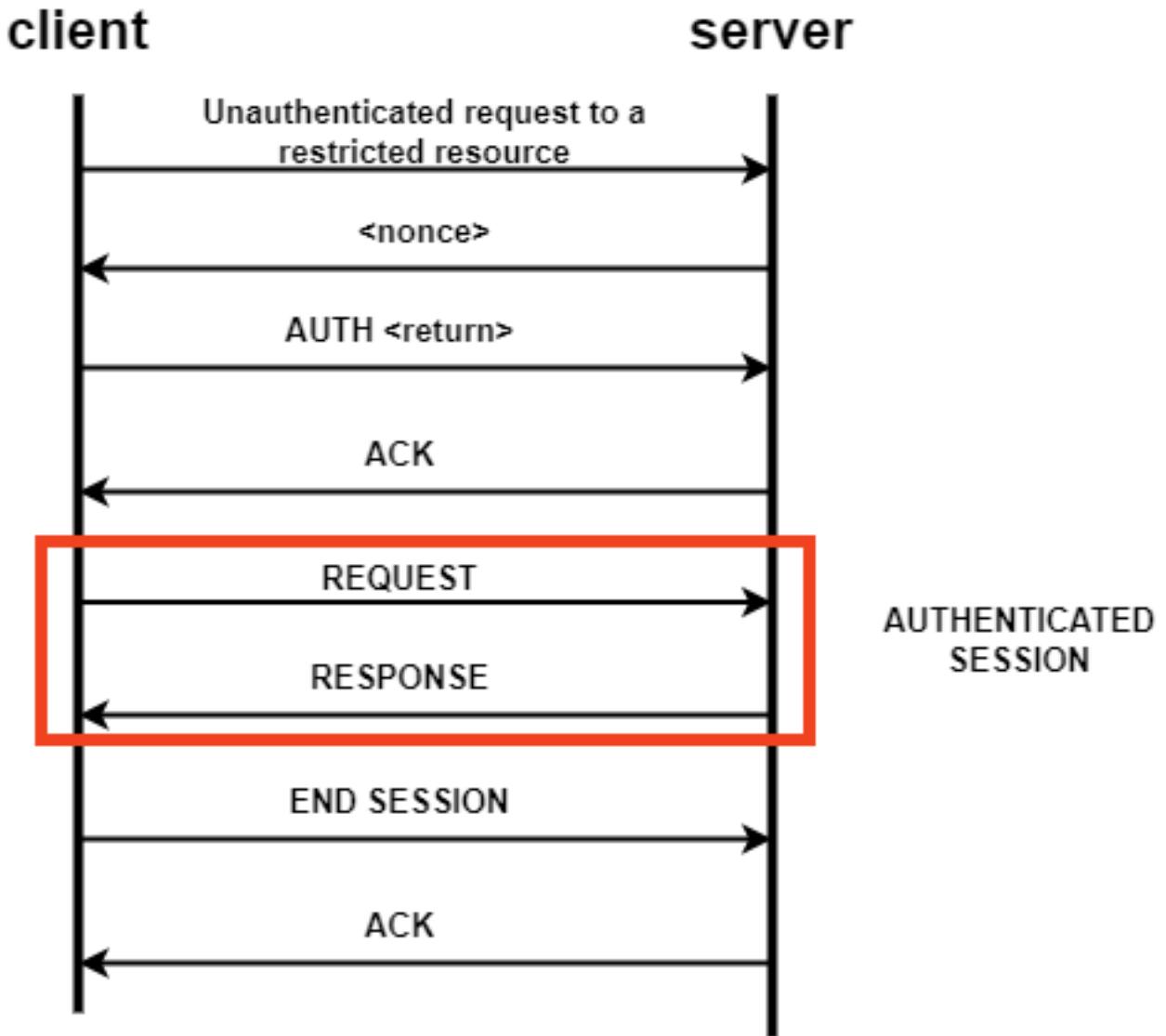
Despite its advantages, HTTP Digest Authentication has some limitations. It is vulnerable to certain attacks, such as dictionary attacks or precomputed hash attacks if weak passwords are used. Additionally, the use of MD5 for hashing has been criticized due to its known vulnerabilities. Therefore, modern implementations of Digest Authentication often use stronger hashing algorithms like SHA-256.

Overall, HTTP Digest Authentication provides a more secure approach to user authentication in HTTP communications compared to Basic Authentication, offering protection against interception, replay attacks, and password storage vulnerabilities.

Flow

- Unauthenticated client requests a restricted resource
- Server generates a random value called a nonce and sends back an HTTP 401 Unauthorized status with a **WWW-Authenticate** header that has a value of **Digest** along with the nonce:
WWW-Authenticate: Digest nonce="23f042f50abcd06ad46fc"
- The **WWW-Authenticate: Basic** header causes the browser to display the username and password prompt
- After entering your credentials, the password is hashed and then sent in the header along with the nonce with each request:
**Authorization: Digest username="username",
nonce="16e30069e45aeb7ab62", response="89549b9c6d93321c52"**
- With the username, the server obtains the password, hashes it along with the nonce, and then verifies that the hashes are the same

Flow



Pros

- More secure than Basic auth since the password is not sent in plain text.
- Easy to implement.
- Supported by all major browsers.

Cons

- User can only be logged out by rewriting the credentials with an invalid one.
- Credentials must be sent with every request.
- Compared to Basic auth, passwords are less secure on the server since bcrypt can't be used.
- Vulnerable to man-in-the-middle attacks.

Packages

- <https://pypi.org/project/django-digest/>

8.2.4 Session-based Auth

Session-based authentication, also known as session-based authentication or session management, is a commonly used method for user authentication and maintaining user sessions in web applications.

In session-based authentication, the process typically involves the following steps:

- **User authentication:** When a user enters their credentials (e.g., user-name and password) on a login page, the server verifies the provided information. If the authentication is successful, the server generates a unique session identifier, often referred to as a session ID.
- **Session creation:** The server creates a session object that is associated with the authenticated user. The session object contains relevant information about the user, such as their user ID or permissions, and is stored on the server side.
- **Session ID assignment:** The server sends the session ID back to the client, typically by setting a session cookie in the user's browser. The session cookie is automatically sent with subsequent requests to the server, allowing the server to identify and associate the requests with the corresponding session.
- **Session validation:** With each subsequent request, the server validates the session by checking if the session ID sent by the client matches an active session on the server. This verification ensures that the user is authenticated and authorized to access the requested resources.
- **Session expiration:** Sessions have a defined duration or can be set to expire after a period of inactivity. When a session expires, the user is typically required to re-authenticate by providing their credentials again to establish a new session.

Session-based authentication provides several advantages:

- **User persistence:** By associating a session with a user, it allows the user to access multiple pages or resources within the application without needing to re-authenticate for each request.
- **Enhanced security:** Session IDs are typically random and unique, making them harder to guess or forge. Additionally, session IDs are stored server-side, reducing the exposure of sensitive user information in the client-side.
- **Flexibility:** Session-based authentication allows for additional session-related functionalities, such as storing user-specific data or implementing session-based permissions.

However, session-based authentication also has certain considerations:

- **Server-side storage:** Session information is stored on the server, requiring server resources to manage and maintain the sessions. This can impact scalability and performance, particularly for applications with high traffic or large numbers of concurrent users.
- **Stateless nature of HTTP:** HTTP itself is stateless, meaning each request is independent. Session-based authentication introduces state by associating requests with sessions. This statefulness needs to be managed and maintained.

It's important to note that session-based authentication has been widely used, but newer authentication methods like token-based authentication (e.g., JSON Web Tokens) or OAuth 2.0 have gained popularity due to their statelessness and scalability advantages.

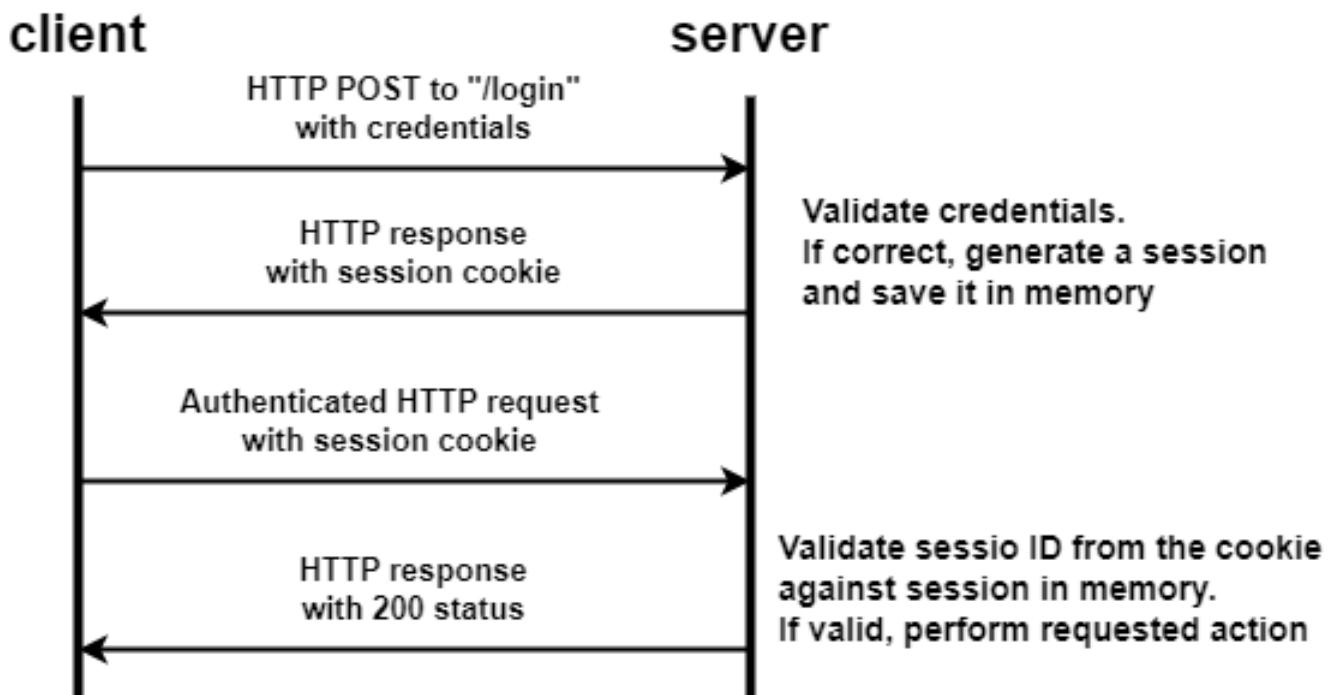
Overall, session-based authentication remains a prevalent approach for managing user authentication and maintaining user sessions in web applications, providing a balance between usability and security.

With session-based auth (or session cookie auth or cookie-based auth), the user's state is stored on the server. It does not require the user to provide a username or a password with each request. Instead, after logging in, the server

validates the credentials. If valid, it generates a session, stores it in a session store, and then sends the session ID back to the browser. The browser stores the session ID as a cookie, which gets sent anytime a request is made to the server.

Session-based auth is stateful. Each time a client requests the server, the server must locate the session in memory in order to tie the session ID back to the associated user.

Flow



Pros

- Faster subsequent logins, as the credentials are not required.
- Improved user experience.
- Fairly easy to implement. Many frameworks (like Django) provide this feature out-of-the-box.

Cons

- It's stateful. The server keeps track of each session on the server-side. The session store, used for storing user session information, needs to be shared across multiple services to enable authentication. Because of this, it doesn't work well for RESTful services, since REST is a stateless protocol.
- Cookies are sent with every request, even if it does not require authentication.
- Vulnerable to CSRF attacks. (Django has a protection built in)

Packages

- <https://docs.djangoproject.com/en/4.2/topics/auth/>

<https://testdriven.io/blog/django-spa-auth/> <https://learndjango.com/tutorials/django-login-and-logout-tutorial>

8.2.5 Token-Based Authentication

Token-based authentication, also known as web token authentication, is an alternative method for user authentication in web applications. It provides a stateless and scalable approach compared to traditional session-based authentication.

Token-based authentication involves the following steps:

- **User authentication:** The user enters their credentials (e.g., user-name and password) on a login page. The server verifies the credentials and, if successful, generates a token.
- **Token generation:** The server generates a token, typically in the form of a JSON Web Token (JWT). The JWT contains encoded information about the user, such as their user ID or roles, and a digital signature to verify the token's authenticity.
- **Token issuance:** The server sends the JWT back to the client, typically as a response to the authentication request. The client stores the token, usually in local storage or a cookie.
- **Token inclusion:** The client includes the token in subsequent requests to the server, typically by adding it to the request headers as an authorization bearer token.
- **Token validation:** Upon receiving a request, the server verifies the token's authenticity by validating the digital signature and decoding the token to extract user information. If the token is valid, the server grants access to the requested resources based on the information contained in the token.

Token-based authentication provides several benefits:

- **Stateless nature:** Tokens are self-contained and carry all the necessary information for authentication. Servers do not need to store session state, making it easier to scale and distribute the authentication process across different servers.
- **Enhanced security:** Tokens are digitally signed, ensuring their integrity and authenticity. Servers can verify the signature to ensure the token hasn't been tampered with.
- **Cross-domain authentication:** Tokens can be issued by an authentication server and used to authenticate requests to multiple services or APIs, even if they are hosted on different domains.
- **Decoupled architecture:** The client includes the token in subsequent requests to the server, typically by adding it to the request headers as an authorization bearer token.

Token-based authentication enables a decoupled architecture, where the authentication server and resource servers can be separate entities. This allows for better separation of concerns and scalability.

However, token-based authentication also has considerations:

- **Token management:** Tokens need to be securely stored on the client-side, either in local storage or as HTTP-only cookies, to prevent unauthorized access or tampering.
- **Token expiration and revocation:** Tokens can have an expiration time, after which they are no longer valid. Revocation of a token typically involves either waiting for it to expire or maintaining a blacklist of revoked tokens.
- **Payload size:** Tokens may contain additional user information, which increases the payload size compared to session IDs. This can impact network bandwidth, particularly in scenarios with high traffic or constrained environments.
- **Decoupled architecture:** The client includes the token in subsequent requests to the server, typically by adding it to the request headers as an authorization bearer token.

Token-based authentication, particularly using JWTs, has gained popularity due to its flexibility, statelessness, and ability to work with various platforms and services. It is commonly used in modern web APIs and single sign-on (SSO) implementations, providing secure and scalable user authentication.

The most commonly used token is a [JSON Web Token](#) (JWT). A JWT consists of three parts:

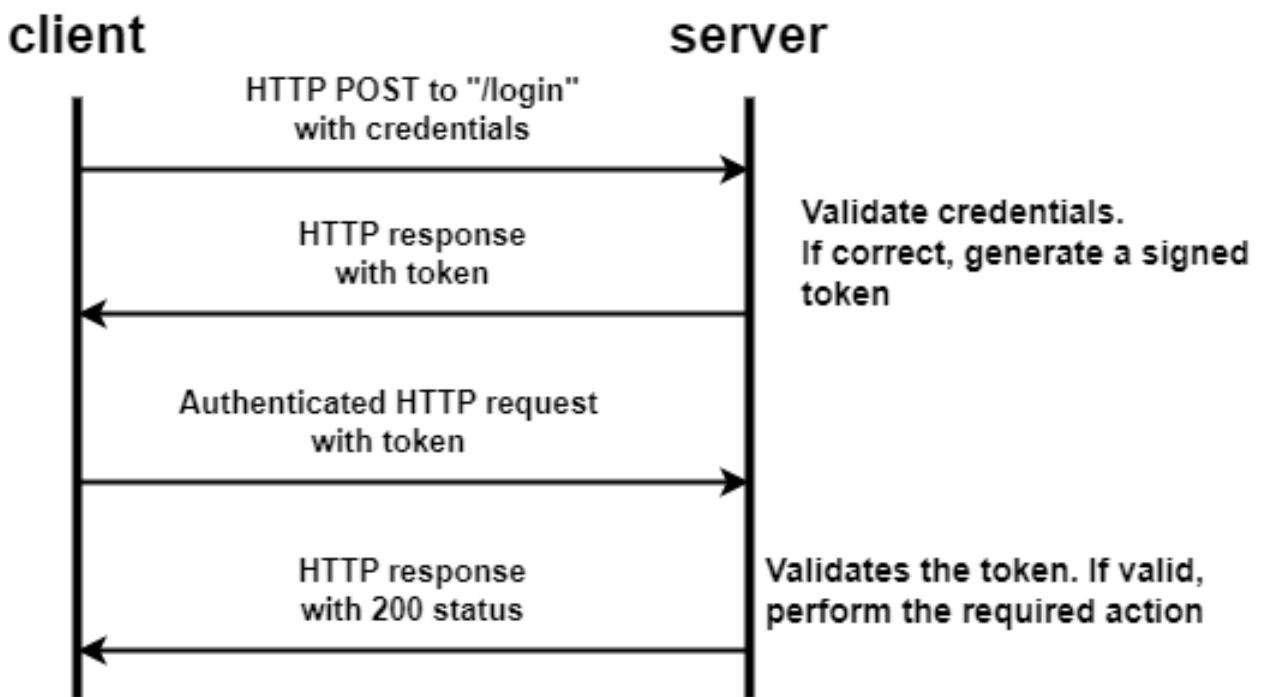
- Header (includes the token type and the hashing algorithm used)
- Payload (includes the claims, which are statements about the subject)
- Signature (used to verify that the message wasn't changed along the way)

All three are base64 encoded and concatenated using a `.` and hashed. Since they are encoded, anyone can decode and read the message. But only authentic users can produce valid signed tokens. The token is authenticated using the Signature, which is signed with a private key.

JSON Web Token (JWT) is a compact, URL-safe means of representing claims to be transferred between two parties. The claims in a JWT are encoded as a JSON object that is used as the payload of a JSON Web Signature (JWS) structure or as the plaintext of a JSON Web Encryption (JWE) structure, enabling the claims to be digitally signed or integrity protected with a Message Authentication Code (MAC) and/or encrypted. - [IETF](#)

Tokens don't need to be saved on the server-side. They can just be validated using their signature. In recent times, token adoption has increased due to the rise of RESTful APIs and Single Page Applications (SPAs).

Flow



Pros

- It's stateless. The server doesn't need to store the token as it can be validated using the signature. This makes the request faster as a database lookup is not required.
- Suited for a microservices architecture, where multiple services require authentication. All we need to configure at each end is how to handle the token and the token secret.

Cons

- Depending on how the token is saved on the client, it can lead to XSS (via localStorage) or CSRF (via cookies) attacks.
- Tokens cannot be deleted. They can only expire. This means that if the token gets leaked, an attacker can misuse it until expiry. Thus, it's important to set token expiry to something very small, like 15 minutes.
- Refresh tokens need to be set up to automatically issue tokens at expiry.
- One way to delete tokens is to create a database for blacklisting tokens. This adds extra overhead to the microservice architecture and introduces state.

Packages

- <https://github.com/SimpleJWT/django-rest-framework-simplejwt>

<https://simpleisbetterthancomplex.com/tutorial/2018/12/19/how-to-use-jwt-authentication-with-django-rest-framework.html>

8.2.6 One Time Passwords

One-Time Passwords (OTP) in web applications provide an additional layer of security for user authentication. OTPs are temporary and can only be used once, making them resistant to replay attacks. There are different methods for implementing OTPs in web applications:

- **Time-based One-Time Password (TOTP):** TOTP is a commonly used OTP method that relies on the current time and a shared secret to generate passwords. The server and the client agree on a shared secret, usually through a QR code or manual entry. The client uses this secret along with the current time to generate a unique password that expires after a short period, typically 30 seconds. The server performs the same calculation and compares the generated OTP with the one provided by the user for authentication.
- **SMS or Email OTP:** In this method, the user provides their phone number or email address during the authentication process. The server sends a one-time password to the user via SMS or email. The user then enters the OTP to authenticate themselves. This method relies on the security of the communication channel (SMS or email) and requires users to have access to their phone or email account during authentication.
- **Push-based OTP:** This method involves a push notification sent to a user's mobile device. When the user initiates the authentication process, a push notification is sent to their device with a prompt to approve or deny the authentication request. Upon approval, the server verifies the response from the user's device and grants access.
- **Hardware Token OTP:** Hardware tokens, such as USB devices or smart cards, can be used to generate OTPs. The token contains a secret key that is used to calculate the OTPs. The user connects the hardware token to their device, and the OTP is generated based on the shared secret and time or a counter.
- **Software-based OTP:** Various software-based OTP solutions, such as Google Authenticator or Microsoft Authenticator, can be used. These applications generate OTPs based on a shared secret and the current time. Users install the authenticator app on their mobile device and associate it with their account. During authentication, the user enters the OTP generated by the app to complete the authentication process.

OTP-based authentication provides an extra layer of security by combining something the user knows (e.g., password) with something temporary and unique (the OTP). It helps protect against password breaches, brute-force attacks, and replay attacks. Implementing OTP in web applications requires a robust and

secure authentication flow, secure transmission channels, and appropriate storage and handling of shared secrets or user contact information.

One time passwords (OTPs) are commonly used as confirmation for authentication. OTPs are randomly generated codes that can be used to verify if the user is who they claim to be. Its often used after user credentials are verified for apps that leverage two-factor authentication.

To use OTP, a trusted system must be present. This trusted system could be a verified email or mobile number.

Modern OTPs are stateless. They can be verified using multiple methods. While there are a few different types of OTPs, Time-based OTPs (TOTPs) is arguably the most common type. Once generated, they expire after a period of time.

Since you get an added layer of security, OTPs are recommended for apps that involve highly sensitive data, like online banking and other financial services.

Flow

The traditional way of implementing OTPs:

- Client sends username and password
- After credential verification, the server generates a random code, stores it on the server-side, and sends the code to the trusted system
- The user gets the code on the trusted system and enters it back on the web app
- The server verifies the code against the one stored and grants access accordingly

How TOTPs work:

- Client sends username and password
- After credential verification, the server generates a random code using a randomly generated seed, stores the seed on the server-side, and sends the code to the trusted system
- The user gets the code on the trusted system and enters it back on the web app
- The server verifies the code against the stored seed, ensures that it has not expired, and grants access accordingly

How OTP agents like Google Authenticator, Microsoft Authenticator, and FreeOTP work:

- Upon registering for Two Factor Authentication (2FA), the server generates a random seed value and sends the seed to the user in the form of unique QR code
- The user scans the QR code using their 2FA application to validate the trusted device
- Whenever the OTP is required, the user checks for the code on their device and enters it on the web app
- The server verifies the code and grants access accordingly

Pros

- Adds an extra layer of protection.
- No danger that a stolen password can be used for multiple sites or services that also implement OTPs.

Cons

- You need to store the seed used for generating OTPs.
- OTP agents like Google Authenticator are difficult to set up again if you lose the recovery code.
- Problems arise when the trusted device is not available (dead battery, network error, etc.). Because of this, a backup device is typically required which adds an additional attack vector.

Packages

- <https://github.com/django-otp/django-otp>

8.2.7 OAuth and OpenID

OAuth and OpenID are related but distinct protocols that are commonly used in the context of web-based authentication and authorization.

OAuth (Open Authorization) is an open standard protocol that allows users to grant limited access to their protected resources on one website (known as the "resource server") to another website or application (known as the "client") without sharing their credentials, such as their username and password. OAuth provides a secure and controlled way for users to delegate access to their resources.

OAuth2 is the successor of OAuth and is widely adopted. It introduces several improvements and enhancements over OAuth 1.0. OAuth2 simplifies the protocol, focuses on specific use cases, and provides a framework for building authorization workflows.

The core components of OAuth2 are:

- **Resource Owner:** The user who owns the protected resources and grants access to them.
- **Client:** The application or website that wants to access the protected resources on behalf of the resource owner. The client obtains authorization to access the resources.
- **Authorization Server:** The server that authenticates the resource owner and issues access tokens to the client.
- **Resource Server:** The server that hosts the protected resources and verifies the access tokens presented by the client to grant or deny access.

The OAuth2 flow typically involves the following steps:

- **Authorization Request:** The client initiates the flow by redirecting the resource owner to the authorization server, requesting access to the protected resources.
- **User Authentication:** The resource owner authenticates themselves with the authorization server.
- **Authorization Grant:** The resource owner provides consent to the client, granting permission to access their protected resources.
- **Access Token Issuance:** The authorization server validates the authentication and authorization, and if successful, issues an access token to the client.
- **Access Resource:** The client presents the access token to the resource server to request access to the protected resources.
- **Resource Server Validation:** The resource server verifies the access token's validity and, if valid, allows the client to access the requested resources.

OpenID, on the other hand, is an authentication protocol built on top of OAuth and allows for secure and standardized user authentication. OpenID enables users

to use their existing accounts from OpenID providers (such as Google, Facebook, or Microsoft) to authenticate themselves on various websites or applications.

The core idea behind OpenID is to establish a decentralized framework for single sign-on (SSO) authentication. With OpenID, users can authenticate once with their OpenID provider and then use that authentication across multiple websites or applications that support OpenID.

OpenID Connect (OIDC) is a widely adopted extension of OpenID that uses OAuth2 as the underlying protocol. OIDC provides additional features, such as user information claims, standardized identity data, and ID token verification.

In summary, OAuth2 is primarily focused on authorization, allowing a client to access protected resources on behalf of a resource owner. OpenID, built on top of OAuth, extends the protocol to provide a decentralized and standardized framework for user authentication across multiple websites or applications, enabling single sign-on capabilities.

Flow

You visit a website that requires you to log in. You navigate to the login page and see a button called "Sign in with Google". You click the button and it takes you to the Google login page. Once authenticated, you're then redirected back to the website that logs you in automatically. This is an example of using OpenID for authentication. It lets you authenticate using an existing account (via an OpenID provider) without the need to create a new account.

The most famous OpenID providers are **Google, Facebook, Twitter, and GitHub**.

After logging in, you navigate to the download service within the website that lets you download large files directly to Google Drive. How does the website get access to your Google Drive? This is where OAuth comes into play. You can grant permissions to access resources on another website. In this case, write access to Google Drive.

Pros

- Improved security.
- Easier and faster log in flows since there's no need to create and remember a username or password.
- In case of a security breach, no third-party damage will occur, as the authentication is passwordless.

Cons

- Your application now depends on another app, outside of your control. If the OpenID system is down, users won't be able to log in.
- People often tend to ignore the permissions requested by OAuth applications.
- Users that don't have accounts on the OpenID providers that you have configured won't be able to access your application. The best approach is to implement both – i.e., username and password and OpenID – and let the user choose.

Packages

- <https://django-allauth.readthedocs.io/>
- <https://django-oauth-toolkit.readthedocs.io/>
- <https://django-oidc-provider.readthedocs.io/>

<https://learndjango.com/tutorials/django-allauth-tutorial>
<https://developer.okta.com/blog/2019/10/21/illustrated-guide-to-oauth-and-oidc>
<https://medium.com/data-rebels/fastapi-google-as-an-external-authentication-provider-3a527672cf33>
<https://realpython.com/flask-google-login/>

8.3 ➤ Implement authentication in Django

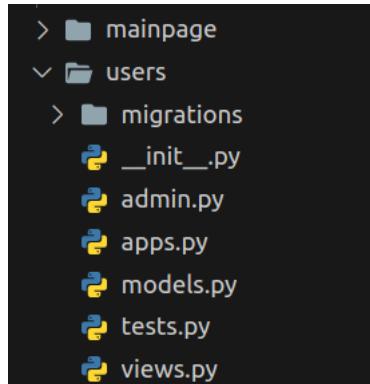
8.3.1 Django website custom user

Since we are going to use a customUser model it will be problematic to add it to the exiting app. This is due to the fact that it already created a migration and setup some dependencies which will no longer be met. The customuser or Customuserbase model has to be created and made before the first migration is done.



occc@occc-sunyorange

```
(venv) occc@occc-VirtualBox:/dj/first_site$ django-admin startapp users  
(venv) occc@occc-VirtualBox:/dj/first_site$
```



Creating a custom user model using AbstractUser

Django recommends that we use a custom user model for all new projects.

Custom User

”Using a custom user model when starting a project If you’re starting a new project, it’s highly recommended to set up a custom user model, even if the default User model is sufficient for you. This model behaves identically to the default user model, but you’ll be able to customize it in the future if the need arises”

[substituting-a-custom-user-model](#)

8.3.2 create user model

users/models.py

```

1  from django.contrib.auth.models import AbstractUser
2  from django.db import models
3
4  class CustomUser(AbstractUser):
5
6      STATUS = (
7          ('student', 'student'),
8          ('faculty', 'faculty'),
9          ('admin', 'admin'),
10     )
11
12     email = models.EmailField(unique=True)
13     status = models.CharField(max_length=100, choices=STATUS, default='regular')
14     description = models.TextField("Description", max_length=600, default='', blank=True)
15
16     def __str__(self):
17         return self.username

```

model choices

To see what items there are in the AbstractUser we can look at the source code. This will be in the env directory under <envdir>/lib/python3.10/site-packages/django/contrib/auth and the file to look at is in models.py.

The Abstractuser model code is:

default AbstratUser model code

```

1
2
3  class AbstractUser(AbstractBaseUser, PermissionsMixin):
4      """
5          An abstract base class implementing a fully featured User model with
6          admin-compliant permissions.
7
8          Username and password are required. Other fields are optional.
9      """
10
11     username_validator = UnicodeUsernameValidator()
12
13     username = models.CharField(
14         _("username"),
15         max_length=150,
16         unique=True,
17         help_text=_(
18             "Required. 150 characters or fewer. Letters, digits and @./+/-/_ only."
19         ),
20         validators=[username_validator],

```

```
1
21     error_messages={
22         "unique": _("A user with that username already exists."),
23     },
24 )
25 first_name = models.CharField(_("first name"), max_length=150, blank=True,
26     ),
27 last_name = models.CharField(_("last name"), max_length=150, blank=True)
28 email = models.EmailField(_("email address"), blank=True)
29 is_staff = models.BooleanField(
30     _("staff status"),
31     default=False,
32     help_text=_("Designates whether the user can log into this admin ↴
33     site."),
34 )
35 is_active = models.BooleanField(
36     _("active"),
37     default=True,
38     help_text=_(
39         "Designates whether this user should be treated as active. "
40         "Unselect this instead of deleting accounts."
41     ),
42 )
43 date_joined = models.DateTimeField(_("date joined"), default=timezone.now,
44     ),
45
46 EMAIL_FIELD = "email"
47 USERNAME_FIELD = "username"
48 REQUIRED_FIELDS = ["email"]
49
50 class Meta:
51     verbose_name = _("user")
52     verbose_name_plural = _("users")
53     abstract = True
54
55     def clean(self):
56         super().clean()
57         self.email = self.__class__.objects.normalize_email(self.email)
58
59     def get_full_name(self):
60         """
61             Return the first_name plus the last_name, with a space in between.
62         """
63         full_name = "%s %s" % (self.first_name, self.last_name)
64         return full_name.strip()
65
66     def get_short_name(self):
67         """
68             Return the short name for the user.
69         """
70         return self.first_name
71
72     def email_user(self, subject, message, from_email=None, **kwargs):
73         """
74             Send an email to this user.
75         """
76         send_mail(subject, message, from_email, [self.email], **kwargs)
```

8.3.3

Enable custom user model authentication

In our project settings.py add:

```
AUTH_USER_MODEL = 'users.CustomUser'
```

```
# Default primary key field type
# https://docs.djangoproject.com/en/4.2/ref/settings/#default-auto-field

DEFAULT_AUTO_FIELD = 'django.db.models.BigAutoField'

AUTH_USER_MODEL = 'users.CustomUser'
```

This will tell Django which class to use for users and not use the default name.

8.3.4

register user model

Add the model to the **admin.py** file in your users app folder.

This is necessary so that when we log into the admin portal to manually add or change user information it will know which information to display.

users/admin.py

```
1 from django.contrib import admin
2 from django.contrib.auth.admin import UserAdmin
3 from .models import CustomUser
4
5
6 from django.contrib import admin
7 from .models import CustomUser
8 from .forms import UserRegistrationForm
9 from django.contrib.auth.admin import UserAdmin
10
```

```
11 # Register your models here.
12 class CustomUserAdmin(UserAdmin):
13     model = CustomUser
14     add_form = UserRegistrationForm
15     fieldsets = (
16         *UserAdmin.fieldsets,
17         (
18             'Other Personal info',
19             {
20                 'fields': (
21                     'status',
22                 )
23             }
24         )
25     )
26
27
28 admin.site.register(CustomUser,CustomUserAdmin)
```

In the above we told the admin page to add the status under a new section called Other personal info.

Register the app:

Register app

```
# Application definition

INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'mainpage',
    'users',
]
```

8.4 > create custom admin form

8.4.1 > register user model

The `form.py` files is not created automatically when the app is created. We have to create this file and add the information below to it.

users/form.py

```

1 from django import forms
2 from django.contrib.auth.forms import ↴
    ↴ UserCreationForm
3 from django.contrib.auth import get_user_model
4 from .models import CustomUser
5
6 class UserRegistrationForm(UserCreationForm):
7     class Meta:
8         model = CustomUser
9         fields = '__all__'
```

8.4.2 > make migrations

Do a test run to see which database changes will be made

```
python manage.py makemigrations --dry-run --verbosity 3
```

```
(venv) occc@occc-VirtualBox:/dj/first_site$ python manage.py makemigrations --dry-run --verbosity 3
Migrations for 'users':
users/migrations/0001_initial.py
- Create model CustomUser
Full migrations file '0001_initial.py':
# Generated by Django 4.2.2 on 2023-06-18 23:28

import django.contrib.auth.models
import django.contrib.auth.validators
from django.db import migrations, models
import django.utils.timezone
```

```

class Migration(migrations.Migration):

    initial = True

    dependencies = [
        ('auth', '0012_alter_user_first_name_max_length'),
    ]

    operations = [
        migrations.CreateModel(
            name='CustomUser',
            fields=[
                ('id', models.BigAutoField(auto_created=True, primary_key=True, serialize=False, verbose_name='ID')),
                ('password', models.CharField(max_length=128, verbose_name='password')),
                ('last_login', models.DateTimeField(blank=True, null=True, verbose_name='last login')),
                ('is_superuser', models.BooleanField(default=False, help_text='Designates that this user has all permissions and can log in and administer the site.')),
                ('username', models.CharField(error_messages={'unique': 'A user with that username already exists.'}, max_length=150, unique=True, validators=[django.contrib.auth.validators.UnicodeUsernameValidator])),
                ('first_name', models.CharField(blank=True, max_length=150, verbose_name='first name')),
                ('last_name', models.CharField(blank=True, max_length=150, verbose_name='last name')),
                ('is_staff', models.BooleanField(default=False, help_text='Designates whether the user can log into this admin site.')),
                ('is_active', models.BooleanField(default=True, help_text='Designates whether this user should be treated as active. Uncheck this instead of deleting accounts.')),
                ('date_joined', models.DateTimeField(default=django.utils.timezone.now, verbose_name='date joined')),
                ('email', models.EmailField(max_length=254, unique=True)),
                ('status', models.CharField(choices=[('student', 'student'), ('faculty', 'faculty'), ('admin', 'admin')], max_length=50)),
                ('description', models.TextField(blank=True, default='', max_length=600, verbose_name='Description')),
                ('groups', models.ManyToManyField(blank=True, help_text='The groups this user belongs to. A user will have direct access to the contents of the groups they belong to.')),
                ('user_permissions', models.ManyToManyField(blank=True, help_text='Specific permissions for this user.')),
            ],
            options={
                'verbose_name': 'user',
                'verbose_name_plural': 'users',
                'abstract': False,
            },
            managers=[
                ('objects', django.contrib.auth.models.UserManager()),
            ],
        ),
    ]

```

(venv) occc@occc-VirtualBox:/dj/first_site\$

```

$ ./manage.py makemigrations
(orange) occc@occc-VirtualBox:/csc108/project1/mywebsite$ python manage.py makemigrations
Migrations for 'users':
  users/migrations/0001_initial.py
    - Create model CustomUser
(orange) occc@occc-VirtualBox:/csc108/project1/mywebsite$
```

Run make migrate.

if you get an error.

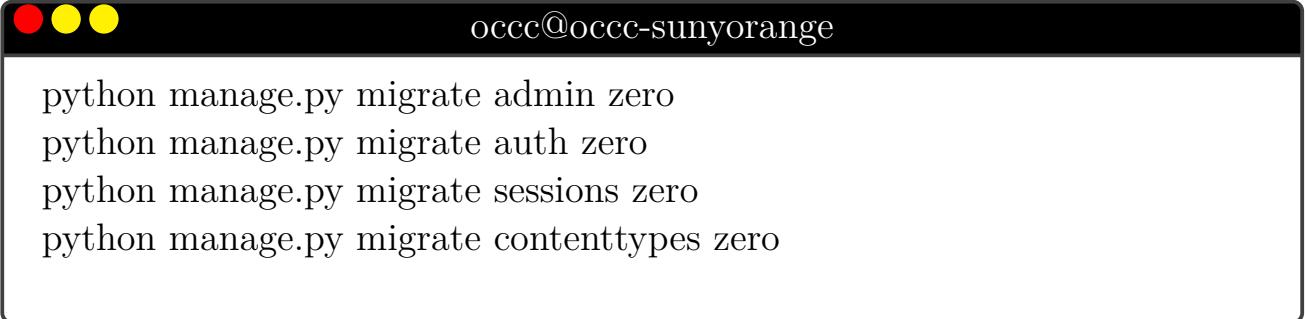
```

(venv) occc@occc-VirtualBox:/dj/first_site$ python manage.py makemigrations
Migrations for 'csc108':
```

```
csc108/migrations/0001_initial.py
- Create model CustomUser
(venv) occc@occc-VirtualBox:/dj/first_site$ python manage.py migrate
Traceback (most recent call last):
File "/dj/first_site/manage.py", line 22, in <module>
    main()
File "/dj/first_site/manage.py", line 18, in main
    execute_from_command_line(sys.argv)
File "/dj/first_site/venv/lib/python3.10/site-packages/django/core/management/__init__.py", line 442,
    utility.execute()
File "/dj/first_site/venv/lib/python3.10/site-packages/django/core/management/__init__.py", line 436,
    self.fetch_command(subcommand).run_from_argv(self.argv)
File "/dj/first_site/venv/lib/python3.10/site-packages/django/core/management/base.py",
    line 412, in run_from_argv
    self.execute(*args, **cmd_options)
File "/dj/first_site/venv/lib/python3.10/site-packages/django/core/management/base.py",
    line 458, in execute
    output = self.handle(*args, **options)
File "/dj/first_site/venv/lib/python3.10/site-packages/django/core/management/base.py",
    line 106, in wrapper
    res = handle_func(*args, **kwargs)
File "/dj/first_site/venv/lib/python3.10/site-packages/django/core/management/commands/migrate.py",
    executor.loader.check_consistent_history(connection)
File "/dj/first_site/venv/lib/python3.10/site-packages/django/db/migrations/loader.py", line 327,
    in check_consistent_history
    raise InconsistentMigrationHistory(
django.db.migrations.exceptions.InconsistentMigrationHistory:
Migration admin.0001_initial is applied before its dependency
csc108.0001_initial on database 'default'.
(venv) occc@occc-VirtualBox:/dj/first_site$
```

Comment out the new app from **INSTALLED_APPS** and comment out the **AUTH_USER_MODEL = 'users.CUstomUser'**

Run:



The screenshot shows a terminal window with a black header bar containing three colored circles (red, yellow, green) on the left and the text "occc@occc-sunyorange" on the right. The main body of the terminal is white and contains the following text:

```
python manage.py migrate admin zero
python manage.py migrate auth zero
python manage.py migrate sessions zero
python manage.py migrate contenttypes zero
```

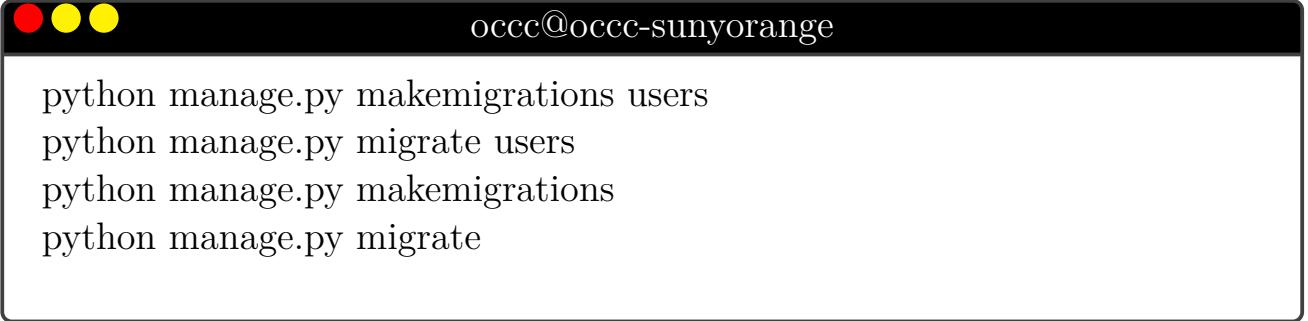
```
(orange) occc@occc-VirtualBox:/csc108/project1/mywebsite$ vi mywebsite/settings.py
(orange) occc@occc-VirtualBox:/csc108/project1/mywebsite$ python manage.py migrate admin zero
Operations to perform:
  Unapply all migrations: admin
Running migrations:
  Rendering model states... DONE
  Unapplying admin.0003_logentry_add_action_flag_choices... OK
  Unapplying admin.0002_logentry_remove_auto_add... OK
  Unapplying admin.0001_initial... OK
(orange) occc@occc-VirtualBox:/csc108/project1/mywebsite$ python manage.py migrate auth zero
Operations to perform:
  Unapply all migrations: auth
Running migrations:
  Rendering model states... DONE
  Unapplying auth.0012_alter_user_first_name_max_length... OK
  Unapplying auth.0011_update_proxy_permissions... OK
  Unapplying auth.0010_alter_group_name_max_length... OK
  Unapplying auth.0009_alter_user_last_name_max_length... OK
  Unapplying auth.0008_alter_user_username_max_length... OK
  Unapplying auth.0007_alter_validators_add_error_messages... OK
  Unapplying auth.0006_require_contenttypes_0002... OK
  Unapplying auth.0005_alter_user_last_login_null... OK
  Unapplying auth.0004_alter_user_username_opts... OK
  Unapplying auth.0003_alter_user_email_max_length... OK
  Unapplying auth.0002_alter_permission_name_max_length... OK
  Unapplying auth.0001_initial... OK
(orange) occc@occc-VirtualBox:/csc108/project1/mywebsite$ python manage.py migrate contenttypes zero
Operations to perform:
  Unapply all migrations: contenttypes
Running migrations:
  Rendering model states... DONE
  Unapplying contenttypes.0002_remove_content_type_name... OK
  Unapplying contenttypes.0001_initial... OK
(orange) occc@occc-VirtualBox:/csc108/project1/mywebsite$ python manage.py migrate sessions zero
Operations to perform:
  Unapply all migrations: sessions
Running migrations:
  Rendering model states... DONE
  Unapplying sessions.0001_initial... OK
(orange) occc@occc-VirtualBox:/csc108/project1/mywebsite$ █
```

```
(venv) occc@occc-VirtualBox:/dj/first_site$ vi firts-site/settings.py
(venv) occc@occc-VirtualBox:/dj/first_site$ python manage.py migrate admin zero
Operations to perform:
Unapply all migrations: admin
Running migrations:
  Rendering model states... DONE
  Unapplying admin.0003_logentry_add_action_flag_choices... OK
  Unapplying admin.0002_logentry_remove_auto_add... OK
  Unapplying admin.0001_initial... OK
(venv) occc@occc-VirtualBox:/dj/first_site$ python manage.py migrate auth zero
Operations to perform:
Unapply all migrations: auth
Running migrations:
  Rendering model states... DONE
  Unapplying auth.0012_alter_user_first_name_max_length... OK
  Unapplying auth.0011_update_proxy_permissions... OK
  Unapplying auth.0010_alter_group_name_max_length... OK
  Unapplying auth.0009_alter_user_last_name_max_length... OK
  Unapplying auth.0008_alter_user_username_max_length... OK
  Unapplying auth.0007_alter_validators_add_error_messages... OK
  Unapplying auth.0006_require_contenttypes_0002... OK
  Unapplying auth.0005_alter_user_last_login_null... OK
  Unapplying auth.0004_alter_user_username_opts... OK
  Unapplying auth.0003_alter_user_email_max_length... OK
  Unapplying auth.0002_alter_permission_name_max_length... OK
  Unapplying auth.0001_initial... OK
(venv) occc@occc-VirtualBox:/dj/first_site$ python manage.py migrate contenttypes zero
```

```
Operations to perform:
Unapply all migrations: contenttypes
Running migrations:
Rendering model states... DONE
Unapplying contenttypes.0002_remove_content_type_name... OK
Unapplying contenttypes.0001_initial... OK
(venv) occc@occc-VirtualBox:/dj/first_site$ python manage.py migrate sessions zero
Operations to perform:
Unapply all migrations: sessions
Running migrations:
Rendering model states... DONE
Unapplying sessions.0001_initial... OK
(venv) occc@occc-VirtualBox:/dj/first_site$ vi mywebsite/settings.py
(venv) occc@occc-VirtualBox:/dj/first_site$ python manage.py migrate users
Operations to perform:
Apply all migrations: users
Running migrations:
Applying contenttypes.0001_initial... OK
Applying contenttypes.0002_remove_content_type_name... OK
Applying auth.0001_initial... OK
Applying auth.0002_alter_permission_name_max_length... OK
Applying auth.0003_alter_user_email_max_length... OK
Applying auth.0004_alter_user_username_opts... OK
Applying auth.0005_alter_user_last_login_null... OK
Applying auth.0006_require_contenttypes_0002... OK
Applying auth.0007_alter_validators_add_error_messages... OK
Applying auth.0008_alter_user_username_max_length... OK
Applying auth.0009_alter_user_last_name_max_length... OK
Applying auth.0010_alter_group_name_max_length... OK
Applying auth.0011_update_proxy_permissions... OK
Applying auth.0012_alter_user_first_name_max_length... OK
Applying users.0001_initial... OK
(venv) occc@occc-VirtualBox:/dj/first_site$ python manage.py makemigrations users
No changes detected in app 'users'
(venv) occc@occc-VirtualBox:/dj/first_site$ python manage.py migrate users
Operations to perform:
Apply all migrations: users
Running migrations:
No migrations to apply.
(venv) occc@occc-VirtualBox:/dj/first_site$ python manage.py makemigrations
No changes detected
(venv) occc@occc-VirtualBox:/dj/first_site$ python manage.py migrate
Operations to perform:
Apply all migrations: admin, auth, contenttypes, sessions, users
Running migrations:
Applying admin.0001_initial... OK
Applying admin.0002_logentry_remove_auto_add... OK
Applying admin.0003_logentry_add_action_flag_choices... OK
Applying sessions.0001_initial... OK
(venv) occc@occc-VirtualBox:/dj/first_site$
```

Go back to settings and uncomment the app and AUTH_USER_MODEL and re-enable the new app

RUN:



```
occc@occc-sunyorange: ~
```

```
python manage.py makemigrations users
python manage.py migrate users
python manage.py makemigrations
python manage.py migrate
```

```
(orange) occc@occc-VirtualBox:/csc108/project1/mywebsite$ vi mywebsite/settings.py
(orange) occc@occc-VirtualBox:/csc108/project1/mywebsite$ python manage.py migrate users
Operations to perform:
  Apply all migrations: users
Running migrations:
  Applying contenttypes.0001_initial... OK
  Applying contenttypes.0002_remove_content_type_name... OK
  Applying auth.0001_initial... OK
  Applying auth.0002_alter_permission_name_max_length... OK
  Applying auth.0003_alter_user_email_max_length... OK
  Applying auth.0004_alter_user_username_opts... OK
  Applying auth.0005_alter_user_last_login_null... OK
  Applying auth.0006_require_contenttypes_0002... OK
  Applying auth.0007_alter_validators_add_error_messages... OK
  Applying auth.0008_alter_user_username_max_length... OK
  Applying auth.0009_alter_user_last_name_max_length... OK
  Applying auth.0010_alter_group_name_max_length... OK
  Applying auth.0011_update_proxy_permissions... OK
  Applying auth.0012_alter_user_first_name_max_length... OK
  Applying users.0001_initial... OK
(orange) occc@occc-VirtualBox:/csc108/project1/mywebsite$ python manage.py makemigrations users
No changes detected in app 'users'
(orange) occc@occc-VirtualBox:/csc108/project1/mywebsite$ python manage.py migrate users
Operations to perform:
  Apply all migrations: users
Running migrations:
  No migrations to apply.
(orange) occc@occc-VirtualBox:/csc108/project1/mywebsite$ python manage.py makemigrations
No changes detected
(orange) occc@occc-VirtualBox:/csc108/project1/mywebsite$ python manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, sessions, users
Running migrations:
  Applying admin.0001_initial... OK
  Applying admin.0002_logentry_remove_auto_add... OK
  Applying admin.0003_logentry_add_action_flag_choices... OK
  Applying sessions.0001_initial... OK
(orange) occc@occc-VirtualBox:/csc108/project1/mywebsite$ █
```

Verify that the new tables were created.

use `python manage.py dbshell` command

```
(orange) occc@occc-VirtualBox:/csc108/project1/mywebsite$ python manage.py dbshell
psql (14.8 (Ubuntu 14.8-0ubuntu0.22.04.1))
SSL connection (protocol: TLSv1.3, cipher: TLS_AES_256_GCM_SHA384, bits: 256, compression: off)
Type "help" for help.

orange=> \dt
              List of relations
 Schema |           Name            | Type | Owner
-----+-----+-----+-----+
 public | auth_group          | table | orangeuser
 public | auth_group_permissions | table | orangeuser
 public | auth_permission      | table | orangeuser
 public | django_admin_log     | table | orangeuser
 public | django_content_type   | table | orangeuser
 public | django_migrations    | table | orangeuser
 public | django_session        | table | orangeuser
 public | users_customuser      | table | orangeuser
 public | users_customuser_groups| table | orangeuser
 public | users_customuser_user_permissions | table | orangeuser
(10 rows)

orange=>
```

```
'
```

```
(venv) occc@occc-VirtualBox:/dj/first_site$ python manage.py dbshell
psql (14.8 (Ubuntu 14.8-0ubuntu0.22.04.1))
SSL connection (protocol: TLSv1.3, cipher: TLS_AES_256_GCM_SHA384, bits: 256, compression: off)
Type "help" for help.

orange=> \dt
List of relations
 Schema |           Name            | Type | Owner
-----+-----+-----+-----+
 public | auth_group          | table | orangeuser
 public | auth_group_permissions | table | orangeuser
 public | auth_permission      | table | orangeuser
 public | django_admin_log     | table | orangeuser
 public | django_content_type   | table | orangeuser
 public | django_migrations    | table | orangeuser
 public | django_session        | table | orangeuser
 public | users_customuser      | table | orangeuser
 public | users_customuser_groups| table | orangeuser
 public | users_customuser_user_permissions | table | orangeuser
(10 rows)

orange=> \q
```

8.5**log back into admin site**

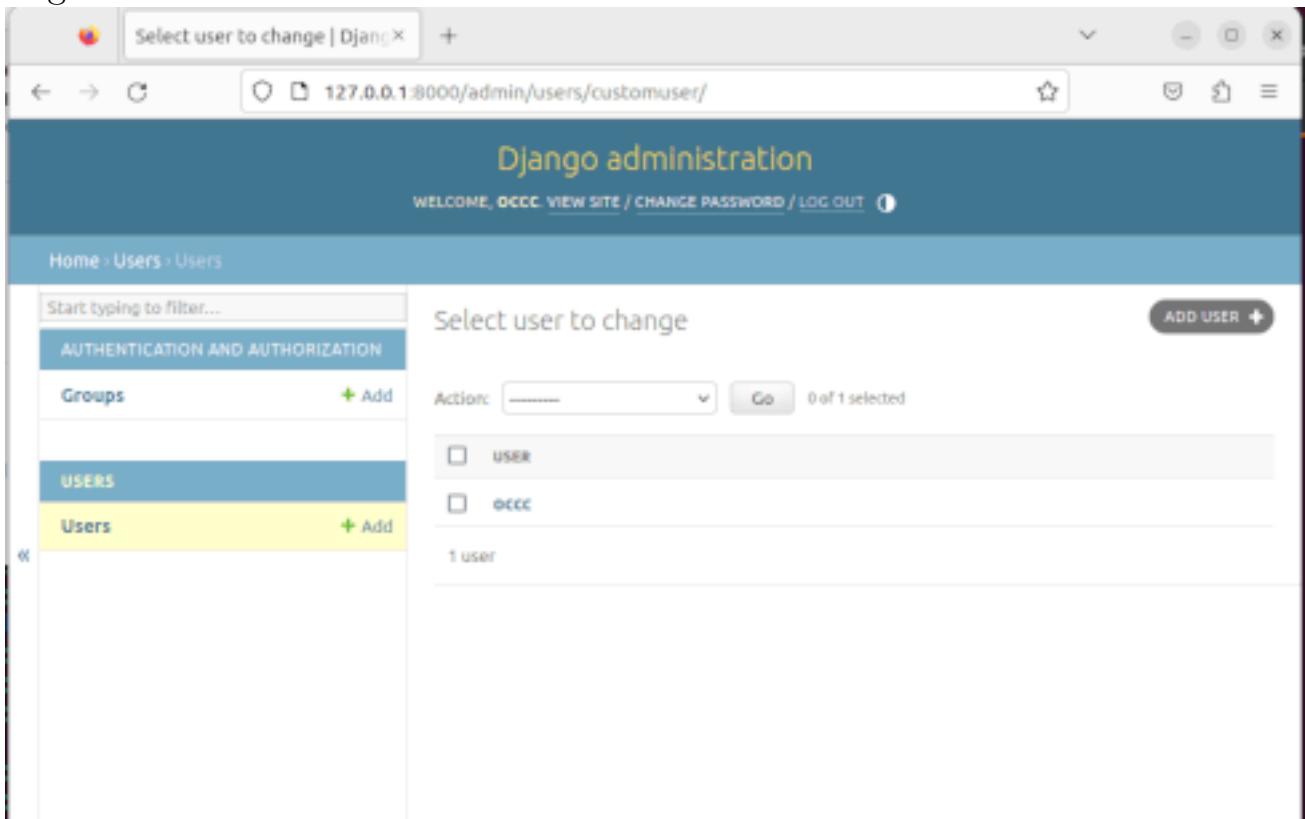
Go to **csc108.sunyorange.edu.local:8000/admin**

If you cannot get back in recreate the superuser account or change the password.

```
(venv) occc@occc-VirtualBox:/dj/first_site$ python manage.py createsuperuser
(venv) occc@occc-VirtualBox:/dj/first_site$ python manage.py changepassword occc
Changing password for user 'occc'
Password:
Password (again):
```

8.6 Add users to the app

Login into the admin site.



Click add user.

fill in the information for password, username,email and status.Then click save.

Add user | Django site admin +

127.0.0.1:8000/admin/users/customuser/add/

Django administration

Welcome, OCCD. View site / Change password / Log out

Home > Users > Users > Add user

Start typing to filter...

AUTHENTICATION AND AUTHORIZATION

Groups

USERS

Users

Add user

Password: orange

Last login: Date: Today

Time: Now

Note: You are 4 hours behind server time.

Superuser status

Designates that this user has all permissions without explicitly assigning them.

Groups:

+

The groups this user belongs to. A user will get all permissions granted to each of their groups. Hold down "Control", or "Command" on a Mac, to select more than one.

User permissions:

admin | log entry | Can add log entry
admin | log entry | Can change log entry
admin | log entry | Can delete log entry
admin | log entry | Can view log entry
auth | group | Can add group
auth | group | Can change group
auth | group | Can delete group
auth | group | Can view group

Specific permissions for this user. Hold down "Control", or "Command" on a Mac, to select more than one.

Username: student1

Required. 150 characters or fewer. Letters, digits and @/./+/-/_ only.

First name: Jane

Last name: Doe

Staff status

Designates whether the user can log into this admin site.

Active

Designates whether this user should be treated as active. Uncheck this instead of deleting accounts.

Date joined: Date: 2023-06-20 Today
Time: 19:19:41 Now

Note: You are 4 hours behind server time.

Email: student1@localhost

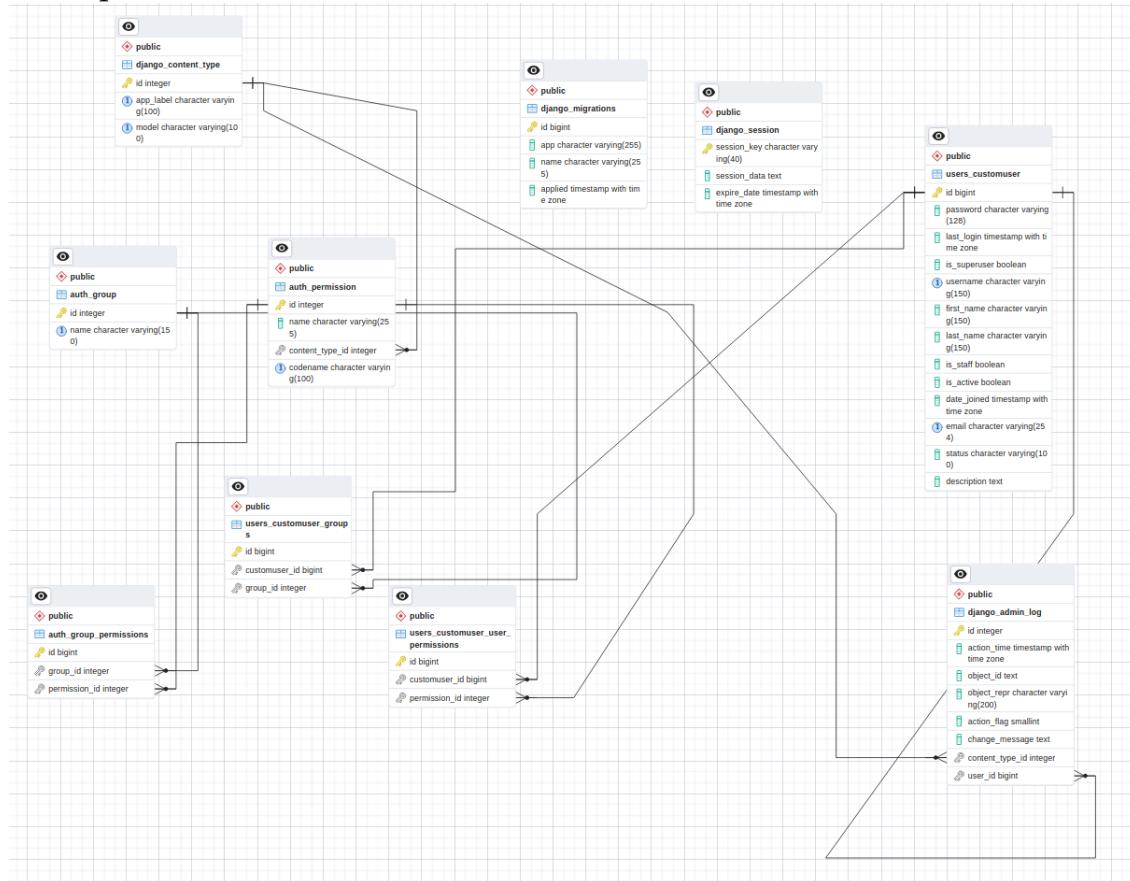
Status: student

Description:

student
faculty
admin

The screenshot shows the Django admin 'Select user to change' page. At the top, a message says 'The user "student1" was added successfully.' Below this, there's a list of users: 'student1' and 'soso'. A button labeled 'ADD USER +' is visible on the right.

Do this two more times but choose faculty and admin next time.
In real production we would add a group to each type of account and restrict their permissions.

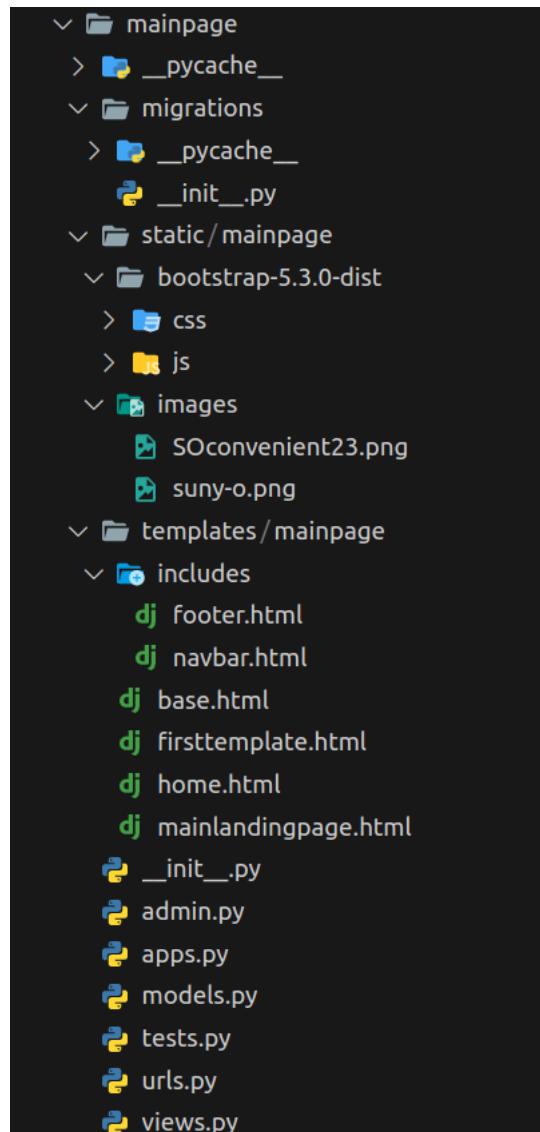


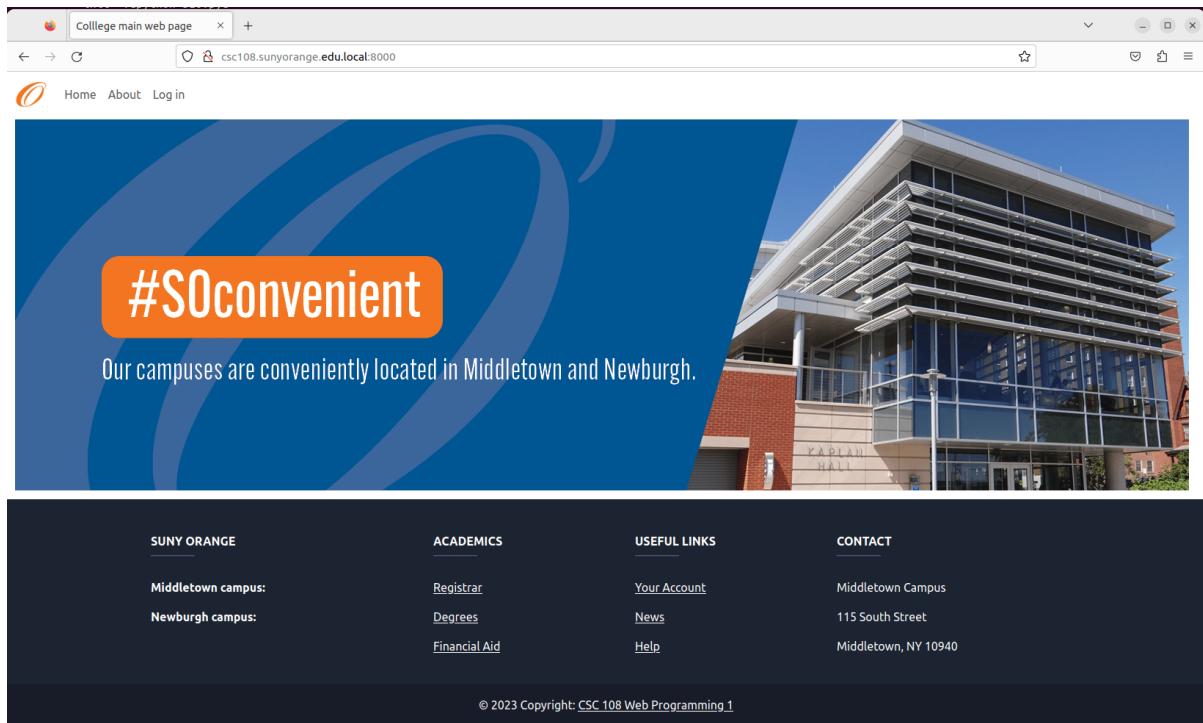
8.7 create an app for each type

We can keep the one main app, but to make editing html and templates simple we will separate each type of account to its own app.

```
occc@occc-sunyorange:~$ (venv) django-admin startapp students
occc@occc-sunyorange:~$ (venv) django-admin startapp faculty
occc@occc-sunyorange:~$ (venv) django-admin startapp registrar
occc@occc-sunyorange:~$ (venv) ls
registrar db.sqlite3 faculty first_site mainpage manage.py students users
occc@occc-sunyorange:~$
```

8.8 ➤ mainsite web page





add error messaging, login and logout to navbar

8.8.1 base html

mainpage/templates/mainpage/base.html

```

1  {% load static %}
2  <!DOCTYPE html>
3  <html lang="en-US">
4    <head>
5      <meta charset="utf-8">
6      <meta name="description" content="CSC 108 Web programming 1">
7      <meta name="keywords" content="Django, Suny Orange, CSC108">
8      <meta name="author" content="Miroslav Krajca">
9      <meta name="Classification" content='College'>
10     <meta name="copyright" content='Miroslav Krajca'>
11     <title>{% block title %}{% endblock title %}</title>
12     <link rel="stylesheet"
13       type="text/css"
14       href="{% static 'mainpage/bootstrap-5.3.0-dist/css/bootstrap.min.css' %}">
15     {% block custom_css %}{% endblock custom_css %}
16   </head>
17   <body>
18     {% include 'mainpage/includes/navbar.html' %}
19     <main role="main" class="container-fluid">
20       <div class="row">
21         <div class="col-lg-12 col-md-12 p-0 col-sm-12">
22           {% include 'mainpage/includes/messaging.html' %}
```

```

24         {% block content %}{% endblock content %}
25     </div>
26     </div>
27 </main>
28 <!-- Optional Javascript -->
29 <script src="{% static 'mainpage/jquery3.7.0/jquery-3.7.0.min.js' %}"></script>
30     ↴ /script>
31 <script src="{% static 'mainpage/bootstrap-5.3.0-dist/js/bootstrap.min.js' %}"></script>
32     ↴ js' %}></script>
33     {% include 'mainpage/includes/footer.html' %}
34 <script>
35     $(document).ready(function() {
36         window.setTimeout(function() {
37             $(".alert").fadeTo(500, 0).slideUp(500, function() {
38                 $(this).remove();
39             });
40         }, 5000);
41     });
42 </script>
43     {% block custom_js %} {% endblock custom_js %}
44 </body>
45 </html>

```

8.8.2 home html

mainpage/templates/mainpage/home.html

```

1  {% extends "mainpage/base.html" %}
2  {% load static %}
3  {% block title %}
4  College Main web page
5  {% endblock title %}
6  {% block custom_css %}
7  {% endblock custom_css %}
8
9
10 {% block content %}
11 
12 {% endblock content %}
13
14 {% block custom_js %} {% endblock custom_js %}

```

8.8.3 include navbar html

mainpage/templates/mainpage/includes/navbar.html

```

1  {% load static %}
2  <nav class="navbar navbar-expand-md navbar-color">
3      <div class="container-fluid">
4          <!-- Navbar brand -->
5          <a class="navbar-brand" href="#">
6              
7          </a>
8          <!-- Navbar Left Side -->
9          <div class="navbar-nav">
11             <a class="nav-item nav-link" href="/">Home</a>
12         </div>
13         <button aria-controls="navbarToggle"
14                 aria-expanded="false"
15                 aria-label="Toggle navigation"
16                 class="navbar-toggler ml-auto float-right btncolor"
17                 data-target="#navbarToggle"
18                 data-toggle="collapse"
19                 type="button">
20             <i class="fas fa-bars fa-2x"></i>
21         </button>
22         <div class="collapse navbar-collapse" id="navbarToggle">
23             <!-- Navbar Left Side -->
24             <div class="navbar-nav mr-auto">
25                 <a class="nav-item nav-link" href="/">About</a>
26             </div>
27             <!-- Navbar Right Side -->
28             <div class="navbar-nav">
29                 {% if user.is_authenticated %}
30                     <a class="nav-item nav-link" href="{% url 'logout' %}">
31                         Log out <i class="fas fa-sign-out-alt"></i>
32                     </a>
33                 {% else %}
34                     <a class="nav-item nav-link" href="{% url 'login' %}">
35                         Log in <i class="fas fa-sign-in-alt"></i>
36                     </a>
37                 {% endif %}
38             </div>
39         </div>
40     </div>
41 </nav>
```

if user.is_authenticated

We can add functions inside the template as long as the function does not take parameters.

In Django, `user.is_authenticated` is an attribute of the user object that checks if the current user is authenticated. In other words, it checks if the user is currently logged in.

user.is_authenticated

```

1  {% if user.is_authenticated %}
2      <p>Welcome, {{ user.username }}. Thanks for logging in.</p>
3  {% else %}
4      <p>Welcome, anonymous user. Please log in.</p>
5  {% endif %}

```

In this example, if the user is authenticated, it displays a welcome message that includes the username of the user. If the user is not authenticated (i.e., not logged in), it displays a different message asking the user to log in.

Note that user in the template context is an instance of django.contrib.auth.models.User (or whatever your user model is if you have a custom user model), and it is automatically added to the context of each RequestContext by the django.contrib.auth.context_processors.auth context processor, which is included by default when you use render() and similar functions.

8.8.4 include footer html

mainpage/templates/mainpage/includes/footer.html

```

1
2  {% load static %}
3  <footer class="text-center text-lg-start text-white fixed-bottom"
4      style="background-color: #1c2331">
5      <!-- Section: Links -->
6      <section class="">
7          <div class="container text-center text-md-start mt-5">
8              <!-- Grid row -->
9              <div class="row mt-3">
10                 <!-- Grid column -->
11                 <div class="col-md-3 col-lg-4 col-xl-3 mx-auto mb-4">
12                     <!-- Content -->
13                     <h6 class="text-uppercase fw-bold">Sunny Orange</h6>
14                     <hr class="mb-4 mt-0 d-inline-block mx-auto"
15                         style="width: 60px;
16                             background-color: #7c4dff;
17                             height: 2px" />
18                     <p>
19                         <strong>Middletown campus:</strong>
20                     </p>
21                     <p>
22                         <strong>Newburgh campus:</strong>
23                     </p>
24                 </div>
25                 <!-- Grid column -->

```

```
26      <!-- Grid column -->
27      <div class="col-md-2 col-lg-2 col-xl-2 mx-auto mb-4">
28          <!-- Links -->
29          <h6 class="text-uppercase fw-bold">Academics</h6>
30          <hr class="mb-4 mt-0 d-inline-block mx-auto"
31              style="width: 60px;
32                  background-color: #7c4dff;
33                  height: 2px" />
34          <p>
35              <a href="#" class="text-white">Registrar</a>
36          </p>
37          <p>
38              <a href="https://catalog.sunyorange.edu/current/programs/index.html"
39                  class="text-white">Degrees</a>
40          </p>
41          <p>
42              <a href="https://sunyorange.edu/financialaid/index.html"
43                  class="text-white">Financial Aid</a>
44          </p>
45      </div>
46      <!-- Grid column -->
47      <!-- Grid column -->
48      <div class="col-md-3 col-lg-2 col-xl-2 mx-auto mb-4">
49          <!-- Links -->
50          <h6 class="text-uppercase fw-bold">Useful links</h6>
51          <hr class="mb-4 mt-0 d-inline-block mx-auto"
52              style="width: 60px;
53                  background-color: #7c4dff;
54                  height: 2px" />
55          <p>
56              <a href="#" class="text-white">Your Account</a>
57          </p>
58          <p>
59              <a href="#" class="text-white">News</a>
60          </p>
61          <p>
62              <a href="#" class="text-white">Help</a>
63          </p>
64      </div>
65      <!-- Grid column -->
66      <!-- Grid column -->
67      <div class="col-md-4 col-lg-3 col-xl-3 mx-auto mb-md-0 mb-4">
68          <!-- Links -->
69          <h6 class="text-uppercase fw-bold">Contact</h6>
70          <hr class="mb-4 mt-0 d-inline-block mx-auto"
71              style="width: 60px;
72                  background-color: #7c4dff;
73                  height: 2px" />
74          <p>Middletown Campus</p>
75          <p>115 South Street</p>
76          <p>Middletown, NY 10940</p>
77      </div>
78      <!-- Grid column -->
79  </div>
80  <!-- Grid row -->
81</div>
```

```

82 </section>
83 <!-- Section: Links -->
84 <!-- Copyright -->
85 <div class="text-center p-3" style="background-color: rgba(0, 0, 0, 0.2)">
86   <br>
87   © 2023 Copyright:
88   <a class="text-white" href="http://csc108.sunyorange.edu.local/">CSC 
89     <br> 108 Web Programming 1</a>
90 </div>
91 <!-- Copyright -->
92 </footer>
93 <!-- Footer -->

```

8.8.5 include messaging html

mainpage/templates.mainpage/includes/messaging.html

```

1  {% if messages %}
2  {% for message in messages %}
3    <div class="container-fluid p-0">
4      <div class="alert {{ message.tags }} alert-dismissible fade show"
5        role="alert">
6        <button type="button"
7          class="btn-close"
8          data-bs-dismiss="alert"
9          aria-label="Close">
10         <span aria-hidden="True">&times;</span>
11       </button>
12       {{ message.message }}
13     </div>
14   </div>
15   {% endfor %}
16 {% endif %}
17

```

[Django messaging](#)

8.8.6 mainpage urls.py

mainpage urls.py

```

1 from django.urls import path
2
3 from . import views
4

```

```
5 urlpatterns = [
6     path("index", views.index, name="index"),
7     path("",views.home, name="home"),
8 ]
```

8.8.7 mainpage views.py

mainpage views.py

```
1 from django.shortcuts import render
2 from django.http import HttpResponseRedirect
3 from django.template import loader
4
5
6 def index(request):
7     """ main landing page for our website """
8     return render(request, "mainpage/home.html")
9
10 def home(request):
11     return render(request, "mainpage/home.html")
```

8.9 user templates and code

8.9.1 user forms.py

users-forms.py

```
1 from django import forms
2 from django.contrib.auth.forms import UserCreationForm
3 from django.contrib.auth import get_user_model
4 from .models import CustomUser
5
6 class UserRegistrationForm(UserCreationForm):
7     class Meta:
8         model = CustomUser
9         fields = '__all__'
```

8.9.2 user admin.py

users-admin.py

```

1 from django.contrib import admin
2 from django.contrib.auth.admin import UserAdmin
3 from .models import CustomUser
4
5
6 from django.contrib import admin
7 from .models import CustomUser
8 from .form import UserRegistrationForm
9 from django.contrib.auth.admin import UserAdmin
10
11 # Register your models here.
12 class CustomUserAdmin(UserAdmin):
13     model = CustomUser
14     add_form = UserRegistrationForm
15     fieldsets = (
16         *UserAdmin.fieldsets,
17         (
18             'Other Personal info',
19             {
20                 'fields': (
21                     'status',
22                 )
23             }
24         )
25     )
26
27
28 admin.site.register(CustomUser,CustomUserAdmin)

```

8.9.3 user models.py

users-models.py

```

1 from django.contrib.auth.models import AbstractUser
2 from django.db import models
3
4 class CustomUser(AbstractUser):
5
6     STATUS = (
7         ('student', 'student'),
8         ('faculty', 'faculty'),
9         ('admin', 'admin'),
10    )
11
12    email = models.EmailField(unique=True)
13    status = models.CharField(max_length=100, choices=STATUS, default='regular')
14    description = models.TextField("Description", max_length=600, default='')

```

```
15     ↴ , blank=True)
16 def __str__(self):
17     return self.username
```

8.9.4 user urls.py

users-urls.py

```
1 from django.urls import path
2 from . import views
3 # from django.contrib.auth import views as auth_views
4
5 urlpatterns = [
6     path('login', views.custom_login, name='login'),
7     path('logout', views.custom_logout, name='logout'),
8 ]
```

8.9.5 project urls.py

project-urls.py

```
1 from django.contrib import admin
2 from django.urls import include, path
3
4 urlpatterns = [
5     path("csc108/", include("mainpage.urls")),
6     path('admin/', admin.site.urls),
7     path("", include("users.urls")),
8 ]
```

8.9.6 user views.py

users-views.py

```
1 #from django.shortcuts import render
2 #from django.contrib.auth.forms import AuthenticationForm
3 from django.shortcuts import render, redirect
4 from django.contrib.auth import get_user_model, login, logout, authenticate
5 from django.contrib import messages
6 from .forms import UserRegistrationForm
7 from django.contrib.auth.decorators import login_required
8 from django.contrib.auth.forms import AuthenticationForm
9 from .models import CustomUser
10 def custom_login(request):
11     if request.user.is_authenticated:
12         return redirect('/')
13
14     if request.method == 'POST':
15         form = AuthenticationForm(None, data=request.POST)
16         if form.is_valid():
17
18             user = authenticate(
19                 username=form.cleaned_data['username'],
20                 password=form.cleaned_data['password'],
21             )
22             if user is not None:
23                 login(request, user)
24                 messages.success(request, f"Hello {user.username}! You have ↴
25                                     ↴ been logged in")
26                 queryset = CustomUser.objects.filter(username=form. ↴
27                                     ↴ cleaned_data['username']).values('status')
28                 user_type = queryset[0]['status']
29                 if(user_type == 'faculty'):
30                     return redirect('faculty')
31                 else:
32                     return redirect('/')
33
34             for error in list(form.errors.values()):
35                 messages.error(request, error)
36
37             form = AuthenticationForm()
38
39             return render(
40                 request=request,
41                 template_name="users/login.html",
42                 context={'form': form}
43             )
44
45 @login_required
46 def custom_logout(request):
47     logout(request)
48     messages.info(request, "Logged out successfully!")
49     return redirect("/")
50 # Create your views here.
```

More on request object.

```
queryset = CustomUser.objects.filter(username=form.cleaned_data['username']).values('status')
```

Django query field lookup sql

Django request object

The Django request object is a core part of how Django handles incoming HTTP requests. It's an instance of the `HttpRequest` class, and it's created by Django each time a client sends a request to a Django web application.

The request object is passed as the first argument to any Django view function, and it contains metadata about the request. This includes things like HTTP headers, the path of the request, HTTP method (GET, POST, etc.), and any data that is sent with the request.

Some but not all attributes. We can dynamically add custom attributes if needed.

- The outer **request.method**: A string representing the HTTP method used in the request. This is typically "GET" or "POST".
- The outer **request.GET**: This is a dictionary-like object that contains all available GET parameters.
- The outer **request.POST**: Similar to request.GET, but contains POST parameters.
- The outer **request.FILES**: A dictionary-like object containing all uploaded files.
- The outer **request.COOKIES**: All available cookies are available from this dictionary-like object.
- The outer **request.path**: A string representing the full path to the requested page, not including the scheme or domain.
- The outer **request.user**: This is an instance of the user model representing the currently logged-in user. If no user is logged in, this will be an instance of AnonymousUser.
- The outer **request.session**: A session object representing the current session.
- The outer **request.META**: A dictionary containing all available HTTP headers. Headers from the client are converted to all uppercase, have 'HTTP_' prepended, and dashes are converted to underscores. For example, a header called User-Agent would be available as request.META['HTTP_USER_AGENT'].

Remember that to access data in request.POST or request.FILES, you should use `request.method == 'POST'` to ensure that the data is only accessed when the method is POST. Trying to access request.POST or request.FILES when the method isn't POST will result in an error.

Django custom authenticationform

request.user.is_authenticated

form.is_valid()

login(request, user)

CustomUser.objects.filter

redirect

form.cleaned_data

function decorator @login_required

@login_required

8.9.7 user templates-user-login.html

users/templates/user/login.html

```
1  {% extends "mainpage/base.html" %} 
2  {% block content %} 
3      <div class="content-section"> 
4          <form method="POST"> 
5              {% csrf_token %} 
6              <fieldset class="form-group"> 
7                  <legend class="border-bottom mb-4">Log In</legend> 
8                  {{ form.as_p }} 
9              </div> 
10             </fieldset> 
11             <div class="form-group"> 
12                 <button class="btn btn-outline-info" type="submit">Login</button> 
13             </div> 
14         </form> 
15         <div class="border-top pt-3"> 
16             <small class="text-muted"> 
17                 Need An Account? <a class="ml-2">Sign Up Now</a> 
18             </small> 
19             <br> 
20             <small class="text-muted"> 
21                 <a href="/">Forgot password?</a> 
22             </small> 
23         </div> 
24     </div> 
25     {% endblock content %}
```

Django form rendering<https://docs.djangoproject.com/en/4.2/ref/csrf/>**8.9.8 user templates-user-logout.html****users/templates/user/logout.html**

```

1  {% extends "mainpage/base.html" %} 
2  {% block content %} 
3      <h2>You have been logged out</h2> 
4      <div class="border-top pt-3" style="min-height:65vh;"> 
5          <small class="text-muted"> 
6              <a href="{% url 'login' %}">Log In Again</a> 
7          </small> 
8      </div> 
9  {% endblock content %}
```

8.10 faculty code**8.10.1 faculty view.py****faculty-views.py**

```

1  from django.shortcuts import render 
2  from django.http import HttpResponseRedirect 
3  from django.template import loader 
4  from django.contrib.auth.decorators import login_required 
5  @login_required 
6  def home(request): 
7      return render(request,"faculty/home.html")
```

8.10.2 faculty faculty-templates-faculty-base.html

faculty/templates/faculty/base.html

```
1  {% load static %}          ↵
2  <!DOCTYPE html>           ↵
3  <html lang="en-US">        ↵
4    <head>                  ↵
5      <meta charset="utf-8">   ↵
6      <meta name="description" content="CSC 108 Web programming 1">   ↵
7      <meta name="keywords" content="Django, Suny Orange, CSC108">   ↵
8      <meta name="author" content="Miroslav Krajca">                 ↵
9      <meta name="Classification" content='College'>                   ↵
10     <meta name="copyright" content='Miroslav Krajca'>                ↵
11     <title>Faculty main page</title>                                ↵
12     <link rel="stylesheet" type="text/css"                                ↵
13       href="{% static 'mainpage/bootstrap-5.3.0-dist/css/bootstrap.min.css' %}">   ↵
14
15   </head>                  ↵
16   <body>                  ↵
17     {% include 'mainpage/includes/navbar.html' %}                      ↵
18     <main role="main" class="container-fluid">                         ↵
19       <div class="row">                                              ↵
20         <div class="col-lg-12 col-md-12 p-0 col-sm-12">             ↵
21           {% include 'mainpage/includes/messaging.html' %}            ↵
22           {% block content %}{% endblock %}                            ↵
23         </div>                                              ↵
24       </div>                                              ↵
25     </main>                                              ↵
26     <!-- Optional Javascript -->                                ↵
27     <script src="{% static 'mainpage/jquery3.7.0/jquery-3.7.0.min.js' %}"></script>   ↵
28     <script src="{% static 'mainpage/bootstrap-5.3.0-dist/js/bootstrap.min.js' %}"></script>   ↵
29     {% include 'mainpage/includes/footer.html' %}                    ↵
30     <script>
31       $(document).ready(function() {
32         window.setTimeout(function() {
33           $(".alert").fadeTo(500, 0).slideUp(500, function() {
34             $(this).remove();
35           });
36         }, 5000);
37       });
38     </script>
39   </body>
40 </html>
```

8.10.3**faculty faculty-templates-faculty-home.html**

faculty/templates/faculty/home.html

```

1  {% extends "faculty/base.html" %} 
2  {% load static %} 
3  {% block content %} 
4      <h3>Welcome Faculty</h3> 
5  {% endblock content %}
```

8.10.4 project urls.py

project urls.py

```

1 from django.contrib import admin
2 from django.urls import include, path
3
4 urlpatterns = [
5     path("csc108/", include("mainpage.urls")),
6     path('admin/', admin.site.urls),
7     path("", include("users.urls")),
8     path("", include('faculty.urls')),
9 ]
```

8.10.5 project settings.py

register app

```

INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'mainpage',
    'users',
    'faculty',
```

]

College main web page

csc108.sunyorange.edu.local:8000/login

Home About Log in

Log In

Username:

Password:

[Login](#)

Need An Account? [Sign Up Now](#)
[Forgot password?](#)

SUNY ORANGE

Middletown campus: [Registrar](#) [Your Account](#)

Newburgh campus: [Degrees](#) [News](#)
[Financial Aid](#) [Help](#)

ACADEMICS

USEFUL LINKS

CONTACT

Middletown Campus
115 South Street
Middletown, NY 10940

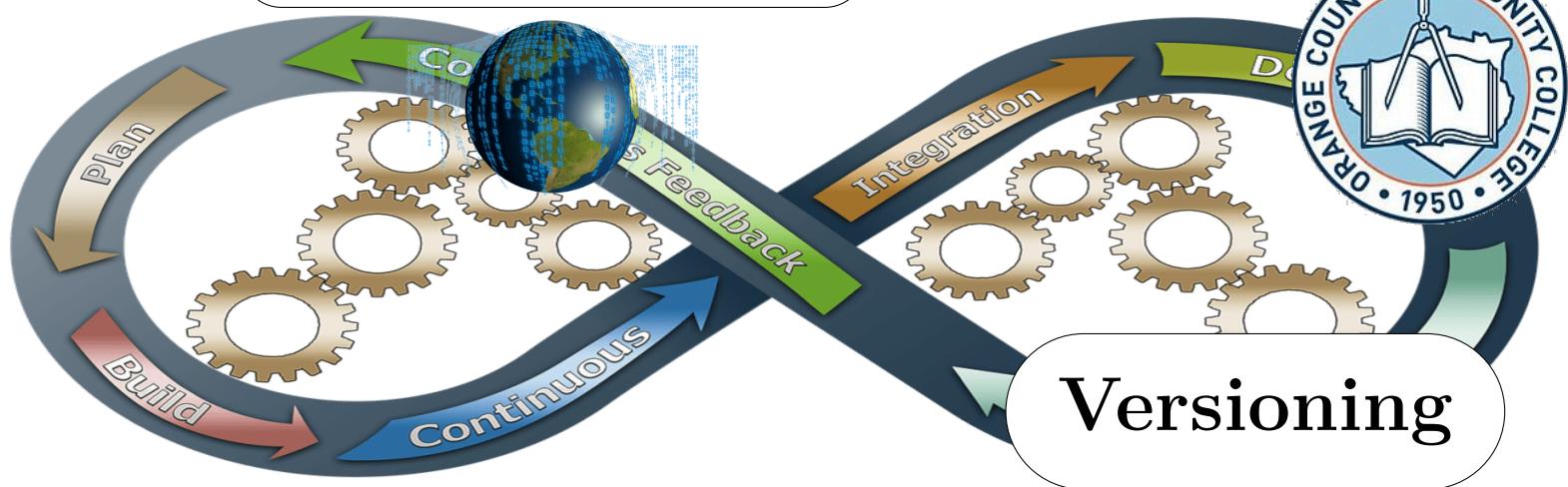
© 2023 Copyright: [CSC 108 Web Programming 1](#)

The screenshot shows a web browser window titled "College main web page". The address bar displays "csc108.sunyorange.edu.local:8000/faculty". The page content includes a logo, navigation links for "Home", "About", and "Log out", and a green notification bar stating "Hello teacher1! You have been logged in". Below this, a large heading says "Welcome Faculty". At the bottom, there is a footer with sections for "SUNY ORANGE", "ACADEMICS", "USEFUL LINKS", and "CONTACT", along with copyright information.

SUNY ORANGE ACADEMICS USEFUL LINKS CONTACT

Middletown campus: [Registrar](#) [Your Account](#) Middletown Campus
Newburgh campus: [Degrees](#) [News](#) 115 South Street
[Financial Aid](#) [Help](#) Middletown, NY 10940

© 2023 Copyright: CSC 108 Web Programming.1



Source code versioning, also known as version control, is the practice of tracking and managing changes to software code. Version control systems are software tools that help software teams manage changes to source code over time.

Version control software keeps track of every modification to the code in a special kind of database. If a mistake is made, developers can turn back the clock and compare earlier versions of the code to help fix the mistake while minimizing disruption to all team members.

There are two main types of version control systems:

- **Centralized Version Control Systems (CVCS):** These systems, like SVN and CVS, use a central server to store all files and enables team collaboration. It works on a single repository to which users can directly access a central server.
- **Distributed Version Control Systems (DVCS):** These systems, like Git and Mercurial, do not necessarily rely on a central server to store all the versions of a project file. In DVCS, every contributor has a complete copy of all the versions of the project files in their local machine.

Git is currently the most popular version control system, notable for its flexibility, performance, and powerful branching and merging capabilities.

Version control is an essential practice for software development teams, especially those adopting Agile development practices or DevOps methodology. It allows developers to work concurrently, provides a way to enforce documentation of code changes, enables reverting back to a previous revision if a bug is found, and allows easy comparison between versions of code files.

9.1

What is git?

Git is a free and open-source distributed version control system that is designed to handle everything from small to very large projects with speed and efficiency. It was created by Linus Torvalds, the creator of Linux, in 2005.

Version control systems are tools that help manage changes to source code over time and allow you to keep track of these changes. Distributed version control systems like Git allow multiple developers to work on a project at the same time.

Here are some key features of Git:

- **Snapshotting and History:** Git takes "snapshots" of your files, meaning it stores a binary snapshot of all your files together as a version of the project. Whenever you commit changes to your project, Git creates a new snapshot and stores a reference to it. You can view your project's history to see who made changes, what changes were made, when they were made, etc.
- **Performance:** Git was designed to work fast, with most operations performed locally, giving it a huge speed advantage on centralized systems that have to communicate with a server.
- **Security:** The integrity of the data stored using Git is assured with a mechanism called checksumming. Each file and commit is checked and a checksum is created for it.
- **Branching and Merging:** Git's branching model allows you to create your own branch to develop features or fix bugs, without affecting the main project (often called the "master" branch). After you're done, you can merge your work back into the master branch.
- **Distributed Development:** Every developer's working copy of the code is also a repository that can contain the full history of all changes, making it possible to work offline and not reliant on a single central server.
- **Staging Area:** Unlike some version control systems, Git has something called a "staging area" or "index". This is an intermediate area where commits can be formatted and reviewed before completing the commit.

Free and Open-Source: Git is released under a very permissive open-source license, which makes it free to use.

The most common way to interact with Git is through the command line, although there are also several graphical user interfaces (GUIs) for Git if you prefer a visual system.

9.2 ➤ git servers and providers

There are several Git providers that offer free services, and they often come with other features, like bug tracking, feature requests, task management, and wikis. Here are some of the most popular:

- **GitHub:** Probably the most famous Git hosting service. In addition to providing Git repository hosting, it offers features like code review, issue tracking, and project management. Free for public and private repositories with unlimited collaborators, but with some limitations on advanced features and private repository size.
- **GitLab:** Offers similar features to GitHub but also provides its software under an open-source license, allowing you to host it on your own servers. GitLab's free tier includes unlimited private repositories and collaborators.
- **Bitbucket:** Owned by Atlassian, it integrates well with other Atlassian products like Jira, Confluence, and Trello. Bitbucket offers free accounts with unlimited public and private repositories, but limits the number of collaborators on private repositories in its free tier.
- **SourceForge:** Known for hosting lots of open-source projects, SourceForge offers a free service where you can host your Git repositories. It also includes issue tracking and project management features.
- **Azure Repos:** Part of Microsoft's Azure DevOps services. It offers unlimited free private Git repositories but limits free users to five collaborators per repository.
- **AWS CodeCommit:** Amazon's entry into this field, part of AWS's suite of developer tools. AWS CodeCommit is free for up to five users, and includes unlimited repositories.

Each of these services has its own unique features and might be a better fit depending on your specific needs, so it's worth checking out each one to see which is the best fit for you.

9.3 install git

We can download git from <https://git-scm.com/>

The screenshot shows the official Git website (<https://git-scm.com/>). At the top, there's a search bar labeled "Search entire site...". Below the header, there's a brief introduction: "Git is a **free and open source** distributed version control system designed to handle everything from small to very large projects with speed and efficiency." To the right of this text is a diagram illustrating the distributed nature of Git, showing multiple repositories connected by bidirectional arrows.

About: The advantages of Git compared to other source control systems.

Documentation: Command reference pages, Pro Git book content, videos and other material.

Downloads: GUI clients and binary releases for all major platforms. A blue arrow points from the "Downloads" section towards the "Community" section.

Community: Get involved! Bug reporting, mailing list, chat, development and more.

A monitor icon on the right displays the latest source release: **2.41.0** (Release Notes: 2023-06-01) with a "Download for Windows" button. Below the monitor are links for "Windows GUIs" and "Tarballs".

At the bottom left, there's a "Pro Git" book icon and a link: "Pro Git by Scott Chacon and Ben Straub is available to [read online](#) for free."

Almost all Linux distributions have git available from their software repositories and we do not need to download the software from git web page.

On windows along with the git software an git bash will be installed. This is simply a MINGW64 environment for windows. It allows us to run a Unix bash shell on microsoft windows.

9.3.1 MINGW64

MinGW-w64 (Minimalist GNU for Windows, 64-bit) is a fork of the earlier MinGW project. It's a free and open source software development environment that allows you to create Microsoft Windows applications.

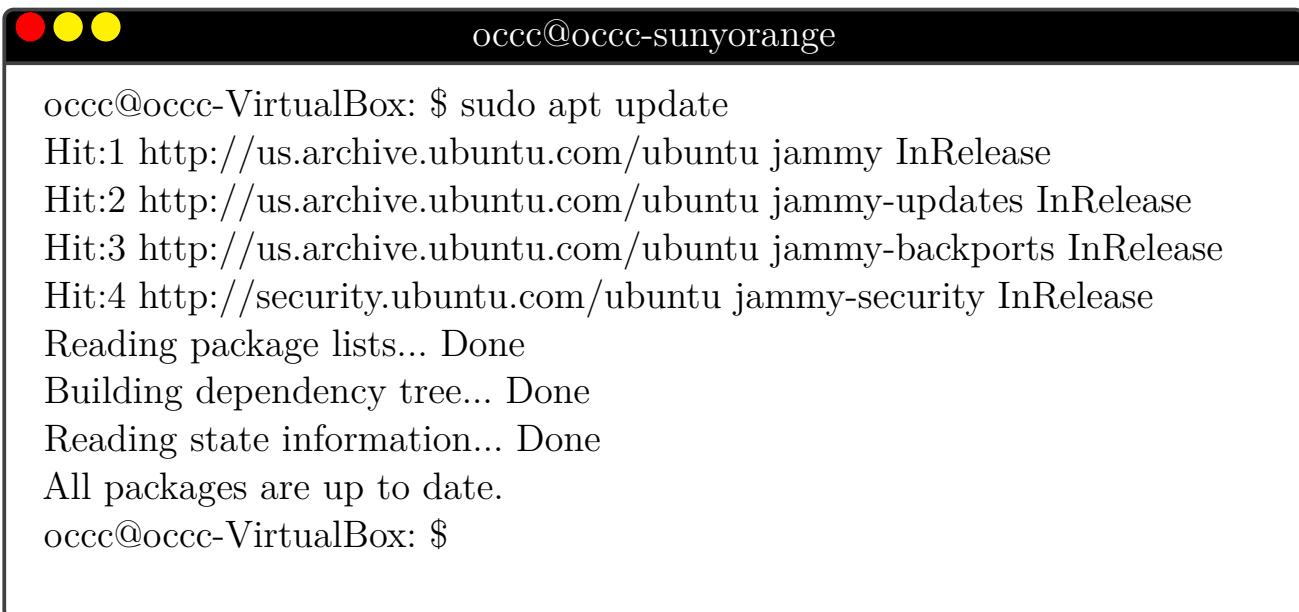
MinGW-w64 includes a port of the GNU Compiler Collection (GCC), GNU Binutils for Windows (assembler, linker, archive manager), a set of freely distributable Windows specific header files and static import libraries which enable the use of the Windows API, a Windows native build of the GNU Project's make utility, and a shell command language interpreter (sh.exe) capable of executing configure scripts.

In other words, it provides a set of tools and compilers that allow developers to generate Windows executables that do not rely on any Unix-like environment, or DLLs (except the standard Microsoft runtimes).

The MinGW-w64 project is notable for its support of a wide range of Windows versions, from Windows XP to Windows 10, and for its inclusion of both 32-bit and 64-bit toolchains.

You might often see MinGW-w64 in action when using Git Bash on Windows. Git Bash is an application that provides Git command line features on Windows. It uses a variant of the MinGW-w64 runtime (msys2) and terminal emulator (mintty), giving you a Unix-like command line environment which is useful for running many programming tools.

9.4 > Linux git install

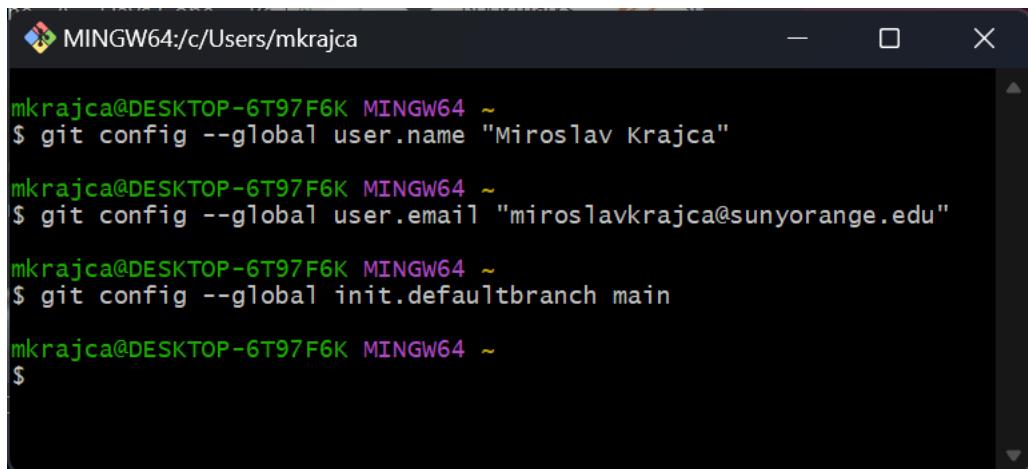


```
occc@occc-VirtualBox: $ sudo apt update
Hit:1 http://us.archive.ubuntu.com/ubuntu jammy InRelease
Hit:2 http://us.archive.ubuntu.com/ubuntu jammy-updates InRelease
Hit:3 http://us.archive.ubuntu.com/ubuntu jammy-backports InRelease
Hit:4 http://security.ubuntu.com/ubuntu jammy-security InRelease
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
All packages are up to date.
occc@occc-VirtualBox: $
```

```
occc@occc-VirtualBox: $ sudo apt install git
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following package was automatically installed and is no longer required:
python3-gunicorn
Use 'sudo apt autoremove' to remove it.
The following additional packages will be installed:
git-man liberror-perl
Suggested packages:
git-daemon-run | git-daemon-sysvinit git-doc git-email git-gui gitk gitweb git-cvs git-mediawiki git-svn
The following NEW packages will be installed:
git git-man liberror-perl
0 upgraded, 3 newly installed, 0 to remove and 0 not upgraded.
Need to get 4,147 kB of archives.
After this operation, 21.0 MB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://us.archive.ubuntu.com/ubuntu jammy/main amd64 liberror-perl all 0.17029-1 [26.5 kB]
Get:2 http://us.archive.ubuntu.com/ubuntu jammy-updates/main amd64 git-man all 1:2.34.1-1ubuntu1.9 [954 kB]
...
Processing triggers for man-db (2.10.2-1) ...
occc@occc-VirtualBox: $
```

9.5 ➤ initial local git setup

9.5.1 ➤ windows git bash setup



```
MINGW64:/c/Users/mkrajca
mkrajca@DESKTOP-6T97F6K MINGW64 ~
$ git config --global user.name "Miroslav Krajca"
mkrajca@DESKTOP-6T97F6K MINGW64 ~
$ git config --global user.email "miroslav.krajca@sunnyorange.edu"
mkrajca@DESKTOP-6T97F6K MINGW64 ~
$ git config --global init.defaultbranch main
mkrajca@DESKTOP-6T97F6K MINGW64 ~
$
```

```
MINGW64:/c/Users/mkrajca
mkrajca@DESKTOP-6T97F6K MINGW64 ~
$ git config --list
credential.helper=manager
diff.astextplain.textconv=astextplain
filter.lfs.clean=git-lfs clean -- %f
filter.lfs.smudge=git-lfs smudge -- %f
filter.lfs.process=git-lfs filter-process
filter.lfs.required=true
http.sslbackend=openssl
http.sslcainfo=C:/Program Files/Git/mingw64/etc/ssl/certs/ca-bundle.crt
core.autocrlf=true
core.fscache=true
core.symlinks=false
pull.rebase=false
credential.https://dev.azure.com.usehttppath=true
init.defaultbranch=master
user.name=Miroslav Krajca
user.email=miroslavkrajca@sunnyorange.edu
init.defaultbranch=main

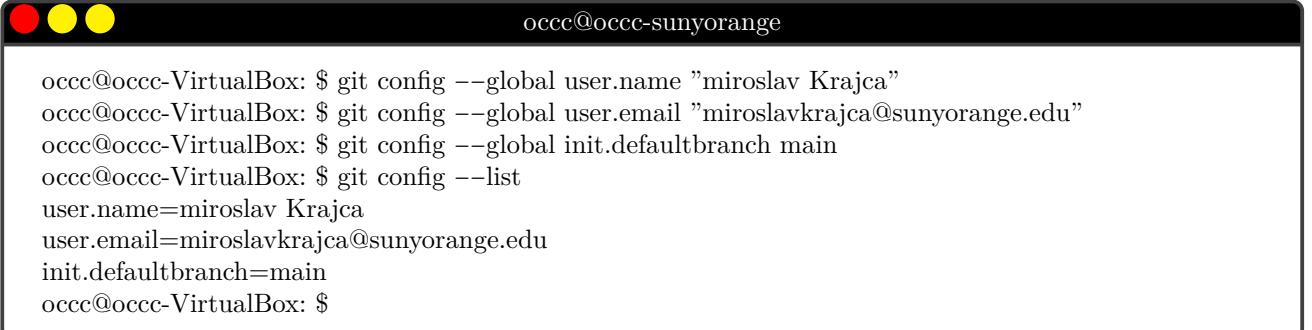
mkrajca@DESKTOP-6T97F6K MINGW64 ~
$
```

```
MINGW64:/c/Users/mkrajca
[user]
    name = Miroslav Krajca
    email = miroslavkrajca@sunnyorange.edu
[init]
    defaultbranch = main
~
(END)
```

```
MINGW64:/c/Users/mkrajca
mkrajca@DESKTOP-6T97F6K MINGW64 ~
$ git config --list --show-origin
file:C:/Program Files/Git/etc/gitconfig credential.helper=manager
file:C:/Program Files/Git/etc/gitconfig diff.astextplain.textconv=astextplain
file:C:/Program Files/Git/etc/gitconfig filter.lfs.clean=git-lfs clean -- %f
file:C:/Program Files/Git/etc/gitconfig filter.lfs.smudge=git-lfs smudge -- %f
file:C:/Program Files/Git/etc/gitconfig filter.lfs.process=git-lfs filter-process
file:C:/Program Files/Git/etc/gitconfig filter.lfs.required=true
file:C:/Program Files/Git/etc/gitconfig http.sslbackend=openssl
file:C:/Program Files/Git/etc/gitconfig http.sslcainfo=C:/Program Files/Git/mingw64/etc/ssl/certs/ca-bundle.crt
file:C:/Program Files/Git/etc/gitconfig core.autocrlf=true
file:C:/Program Files/Git/etc/gitconfig core.fscache=true
file:C:/Program Files/Git/etc/gitconfig core.symlinks=false
file:C:/Program Files/Git/etc/gitconfig pull.rebase=false
file:C:/Program Files/Git/etc/gitconfig credential.https://dev.azure.com.usehttppath=true
file:C:/Program Files/Git/etc/gitconfig init.defaultbranch=master
file:C:/Users/mkrajca/.gitconfig user.name=Miroslav Krajca
file:C:/Users/mkrajca/.gitconfig user.email=miroslavkrajca@sunnyorange.edu
file:C:/Users/mkrajca/.gitconfig init.defaultbranch=main

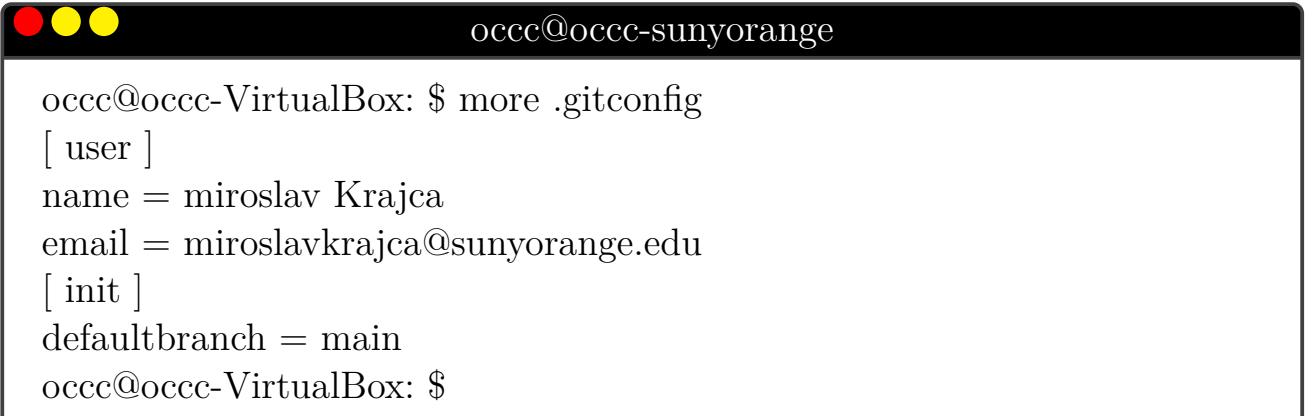
mkrajca@DESKTOP-6T97F6K MINGW64 ~
$ |
```

9.5.2 linux setup

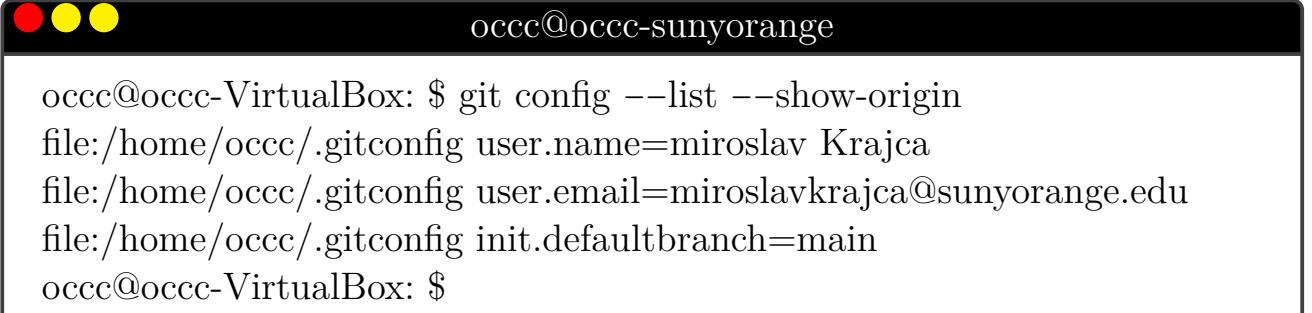


```
occc@occc-VirtualBox: $ git config --global user.name "miroslav Krajca"
occc@occc-VirtualBox: $ git config --global user.email "miroslavkrajca@sunyorange.edu"
occc@occc-VirtualBox: $ git config --global init.defaultbranch main
occc@occc-VirtualBox: $ git config --list
user.name=miroslav Krajca
user.email=miroslavkrajca@sunyorange.edu
init.defaultbranch=main
occc@occc-VirtualBox: $
```

The contents of this setup are stored in Linux in your home directory in a file **.gitconfig**



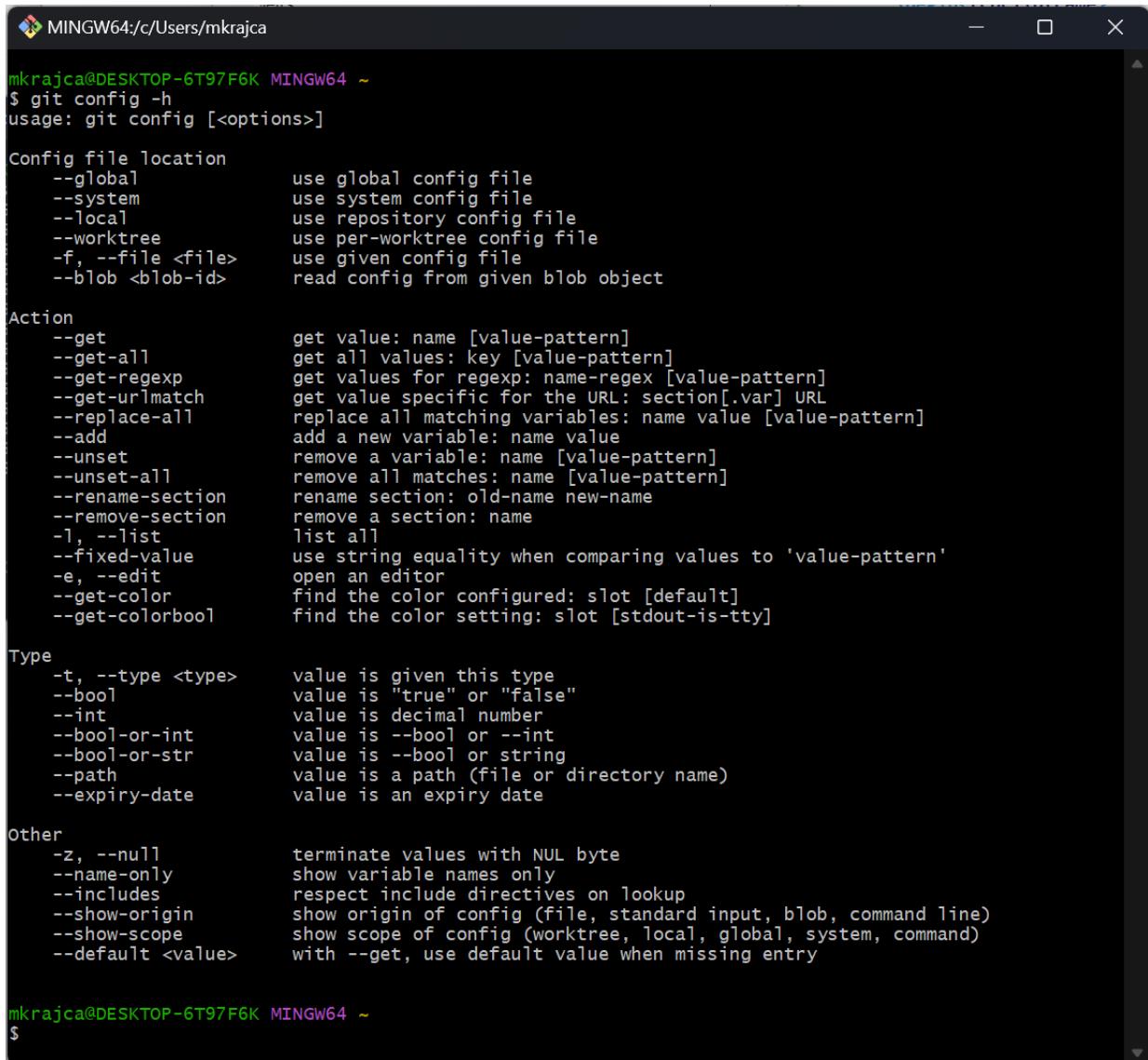
```
occc@occc-VirtualBox: $ more .gitconfig
[ user ]
name = miroslav Krajca
email = miroslavkrajca@sunyorange.edu
[ init ]
defaultbranch = main
occc@occc-VirtualBox: $
```



```
occc@occc-VirtualBox: $ git config --list --show-origin
file:/home/occc/.gitconfig user.name=miroslav Krajca
file:/home/occc/.gitconfig user.email=miroslavkrajca@sunyorange.edu
file:/home/occc/.gitconfig init.defaultbranch=main
occc@occc-VirtualBox: $
```

9.6 ➤ getting help with git commands

git config -h



```

MINGW64:/c/Users/mkrajca
mkrajca@DESKTOP-6T97F6K MINGW64 ~
$ git config -h
usage: git config [<options>]

Config file location
--global           use global config file
--system           use system config file
--local            use repository config file
--worktree         use per-worktree config file
-f, --file <file> use given config file
--blob <blob-id>  read config from given blob object

Action
--get              get value: name [value-pattern]
--get-all          get all values: key [value-pattern]
--get-regexp       get values for regexp: name-regex [value-pattern]
--get-urlmatch    get value specific for the URL: section[.var] URL
--replace-all     replace all matching variables: name value [value-pattern]
--add              add a new variable: name value
--unset            remove a variable: name [value-pattern]
--unset-all        remove all matches: name [value-pattern]
--rename-section   rename section: old-name new-name
--remove-section   remove a section: name
-l, --list          list all
--fixed-value     use string equality when comparing values to 'value-pattern'
-e, --edit          open an editor
--get-color        find the color configured: slot [default]
--get-colorbool   find the color setting: slot [stdout-is-tty]

Type
-t, --type <type>  value is given this type
--bool             value is "true" or "false"
--int              value is decimal number
--bool-or-int     value is --bool or --int
--bool-or-str     value is --bool or string
--path             value is a path (file or directory name)
--expiry-date     value is an expiry date

Other
-z, --null          terminate values with NUL byte
--name-only        show variable names only
--includes         respect include directives on lookup
--show-origin      show origin of config (file, standard input, blob, command line)
--show-scope       show scope of config (worktree, local, global, system, command)
--default <value>  with --get, use default value when missing entry

mkrajca@DESKTOP-6T97F6K MINGW64 ~
$ |

```

git help config



```

MINGW64:/c/Users/mkrajca
mkrajca@DESKTOP-6T97F6K MINGW64 ~
$ git help config
mkrajca@DESKTOP-6T97F6K MINGW64 ~
$ |

```

opens a webpage with the help file

git-config(1) Manual Page

NAME

git-config - Get and set repository or global options

SYNOPSIS

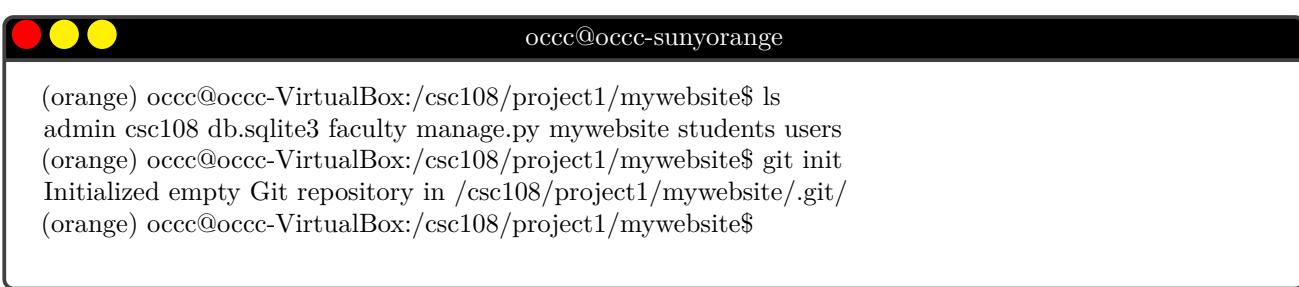
```
git config [<file-option>] [--type=<type>] [--fixed-value] [--show-origin] [--show-scope] [-z|--null] <name> [<value> [<value-pattern>]]
git config [<file-option>] [--type=<type>] --add <name> <value>
git config [<file-option>] [--type=<type>] [--fixed-value] --replace-all <name> <value> [<value-pattern>]
git config [<file-option>] [--type=<type>] [--show-origin] [--show-scope] [-z|--null] [--fixed-value] --get <name> [<value-pattern>]
git config [<file-option>] [--type=<type>] [--show-origin] [--show-scope] [-z|--null] [--fixed-value] --get-all <name> [<value-pattern>]
git config [<file-option>] [--type=<type>] [--show-origin] [--show-scope] [-z|--null] [--fixed-value] [--name-only] --get-regexp <name-regex> [<value-pattern>]
git config [<file-option>] [--type=<type>] [-z|--null] --get-urlmatch <name> <URL>
git config [<file-option>] [--fixed-value] --unset <name> [<value-pattern>]
git config [<file-option>] [--fixed-value] --unset-all <name> [<value-pattern>]
git config [<file-option>] --rename-section <old-name> <new-name>
git config [<file-option>] --remove-section <name>
```

9.7 ➤ start git with our project

Change into our project location on the file system.

The first command is to initialize an empty git repository.

git init



```
(orange) occc@occc-VirtualBox:/csc108/project1/mywebsite$ ls
admin db.sqlite3 faculty manage.py mywebsite students users
(orange) occc@occc-VirtualBox:/csc108/project1/mywebsite$ git init
Initialized empty Git repository in /csc108/project1/mywebsite/.git/
(orange) occc@occc-VirtualBox:/csc108/project1/mywebsite$
```

The **git init** command is used to initialize a new repository in Git. When you use this command, Git creates a new subdirectory named **.git** in your current directory, which contains all the necessary metadata for the new repository. This metadata includes subdirectories for **objects**, **refs/heads**, **refs/tags**, and

template files. In other words, it sets up the necessary Git infrastructure for version controlling your project.

```
(orange) occc@occc-VirtualBox:/csc108/project1/mywebsite$ tree .git
.git
├── branches
├── config
├── description
└── HEAD
├── hooks
│   ├── applypatch-msg.sample
│   ├── commit-msg.sample
│   ├── fsmonitor-watchman.sample
│   ├── post-update.sample
│   ├── pre-applypatch.sample
│   ├── pre-commit.sample
│   ├── pre-merge-commit.sample
│   ├── prepare-commit-msg.sample
│   ├── pre-push.sample
│   ├── pre-rebase.sample
│   ├── pre-receive.sample
│   └── push-to-checkout.sample
        └── update.sample
├── info
│   └── exclude
└── objects
    ├── info
    └── pack
└── refs
    ├── heads
    └── tags

9 directories, 17 files
(orange) occc@occc-VirtualBox:/csc108/project1/mywebsite$ █
```

The directory where you run **git init** becomes the root of the working directory for your project. From that point onwards, you can start using other Git commands to track changes, commit changes, create branches, and more.

git status

The **git status** command in Git is used to display the state of the working directory and the staging area. It lets you see which changes have been staged, which haven't, and which files aren't being tracked by Git.

When you run this command, Git will show you several pieces of information:

- **Current branch:** This tells you what branch you're currently on.
- **Changes to be committed:** If you've added changes to the staging area but haven't committed them yet, they will appear here. This is often referred to as "changes staged for commit."
- **Changes not staged for commit:** If you've made changes to a file that's already tracked by Git, but haven't added those changes to the staging area, they will appear here.
- **Untracked files:** If there are any files in your working directory that Git isn't tracking (usually new files you've created), they will appear here.

```
(orange) occc@occc-VirtualBox:/csc108/project1/mywebsite$ git status
On branch main
No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
    admin/
    csc108/
    db.sqlite3
    faculty/
    manage.py
    mywebsite/
    students/
    users/

nothing added to commit but untracked files present (use "git add" to track)
(orange) occc@occc-VirtualBox:/csc108/project1/mywebsite$ █
```

git add

The **git add** command in Git is used to add changes in the working directory to the staging area, preparing them for the next commit. In other words, git add tells Git that you want to include the updates to a particular file or set of files in the next commit.

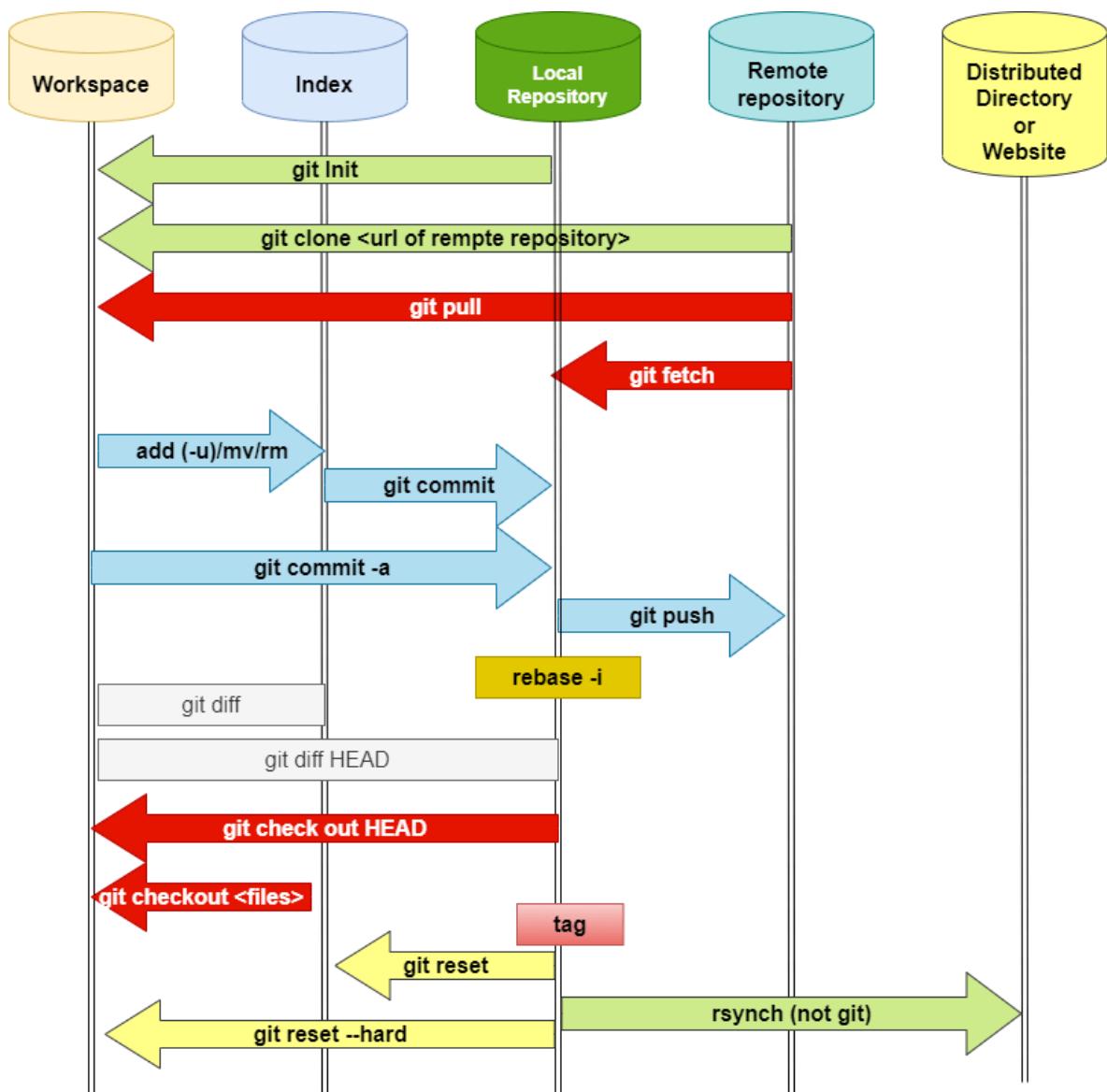
```
(orange) occc@occc-VirtualBox:/csc108/project1/mywebsite$ touch testfile.txt
(orange) occc@occc-VirtualBox:/csc108/project1/mywebsite$ echo "this is a test" > testfile.txt
(orange) occc@occc-VirtualBox:/csc108/project1/mywebsite$ more testfile.txt
this is a test
(orange) occc@occc-VirtualBox:/csc108/project1/mywebsite$ git add testfile.txt
(orange) occc@occc-VirtualBox:/csc108/project1/mywebsite$ git status
On branch main

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
    new file:  testfile.txt

Untracked files:
  (use "git add <file>..." to include in what will be committed)
    admin/
    csc108/
    db.sqlite3
    faculty/
    manage.py
    mywebsite/
    students/
    users/

(orange) occc@occc-VirtualBox:/csc108/project1/mywebsite$ █
```



The above diagram shows us how git works on a high level overview. The staging part is that it add the metadata to git (creates an index about the file)

but it does not actually make a permanent record of the file and its changes. We can look at this in terms of an analogy like shopping on a website. The add portion is us adding something to the cart. Commit is the checkout portion of the shopping. We can add something to a cart and remove it, but it does not charge our payment method.

You can also add all changes in the working directory to the staging area by using a period or an asterisk:

```
git add .
```

```
git add *
```

As noted above "git add" doesn't affect the repository in an irreversible way; changes are not actually recorded until you run "git commit". This gives you a chance to prepare a commit by staging only certain changes, or by modifying files further before staging and committing them.

If you have untracked files (i.e., new files that haven't been added to the repository yet), git add will start tracking those files.

Also, if you've modified a file since the last commit and you run git add on it, the latest changes will be staged.

The git add command doesn't just look at the current state of your files, it takes a snapshot of all the changes you've staged when you run git commit. This makes it possible to stage parts of changes, and make multiple additions before you do a commit.

```
git rm
```

```
(orange) occc@occc-VirtualBox:/csc108/project1/mywebsite$ git rm --cached testfile.txt
rm 'testfile.txt'
(orange) occc@occc-VirtualBox:/csc108/project1/mywebsite$ git status
On branch main

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
    admin/
    csc108/
    db.sqlite3
    faculty/
    manage.py
    mywebsite/
    students/
    testfile.txt
    users/

nothing added to commit but untracked files present (use "git add" to track)
(orange) occc@occc-VirtualBox:/csc108/project1/mywebsite$ █
```

gitignore

Before we add out files to the repo we have to be careful about adding files that should not be tracked. One some files are not really useful for our project and are recreated every time we run our code. **Most important** some files contain extremely sensitive information which if someone got a hold of will pose a major security risk. We have some files that may contain user names, passwords, internal ips, encryption keys, etc ...

We can setup an ignore file which will prevent git from adding or doing any commits with those sensitive documents.

.gitignore is a special file that you can create in your Git repository to tell Git which files or directories to ignore and not track. ensitive data, from being committed into the Git repository.

Each line in the **.gitignore** file specifies a pattern. When Git determines whether to ignore a file, it checks each file and directory against these patterns.

Here are some ways that you can specify patterns in .gitignore:

- **Blank lines** or lines starting with `#` are ignored. This allows you to include comments in your `.gitignore` file.
- **Standard glob patterns** work, such as `*.log` would ignore all files with the `.log` extension.
- **Directories** can be specified by including a slash at the end, like `mydirectory/`.
- **Negation** of a pattern can be done by starting it with an exclamation point (`!`). For instance, `!important.log` would track the file `important.log` even if `*.log` is specified to be ignored.
- **The `/` symbol at the start of a pattern** matches files only in the repository root. So, `/foo` would match a file named `foo` in the repository root, but not a file named `foo` in a subdirectory.
- **The `**` symbol** can be used as a wildcard to match any number of directories. For instance, `**/foo` would match a file or directory named `foo` anywhere in the repository.

Example of what a `.gitignore` file might look like:

```
# Ignore all log files
*.log

# But do track important.log, even though you're ignoring .log files above
!important.log

# Ignore all files in the build/ directory
build/

# Ignore all .txt files in the doc/ directory and any of its subdirectories
doc/**/*.*txt
```

Once you've defined the files and directories to ignore, you need to commit the `.gitignore` file into your repository. This way, other people working on the project will share the same ignore rules.

Note: If a file is already tracked by Git before it was added to the `.gitignore`, the file will continue to be tracked even after it's added to the `.gitignore`. To stop tracking such a file, you need to manually untrack it using the `git rm --cached`

<file> command.

Sample of github ignore files:

<https://github.com/github/gitignore>

Note these are generic and should not be considered for all case, but it might give you a good starting point.

The screenshot shows the GitHub repository 'gitignore' at the URL <https://github.com/github/gitignore>. The repository has 3,528 commits and 8 branches. The 'About' section includes links to Readme, CCO-1.0 license, Code of conduct, Security policy, Activity, and Report repository. The 'Releases' section shows 'No releases published'. The 'Packages' section is empty.

Create our .gitignore file.

```
(orange) occc@occc-VirtualBox:/csc108/project1/mywebsite$ touch .gitignore
(orange) occc@occc-VirtualBox:/csc108/project1/mywebsite$ vi .gitignore
(orange) occc@occc-VirtualBox:/csc108/project1/mywebsite$ more .gitignore
#ignore all sqlite database files
*.sqlite3

(orange) occc@occc-VirtualBox:/csc108/project1/mywebsite$
```

verify gitignore

Git status before placing .gitignore

```
(orange) occc@occc-VirtualBox:/csc108/project1/mywebsite$ git status
On branch main

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
    admin/
    csc108/
    db.sqlite3
    faculty/
    manage.py
    mywebsite/
    students/
    testfile.txt
    users/

nothing added to commit but untracked files present (use "git add" to track)
```

Git status after .gitignore

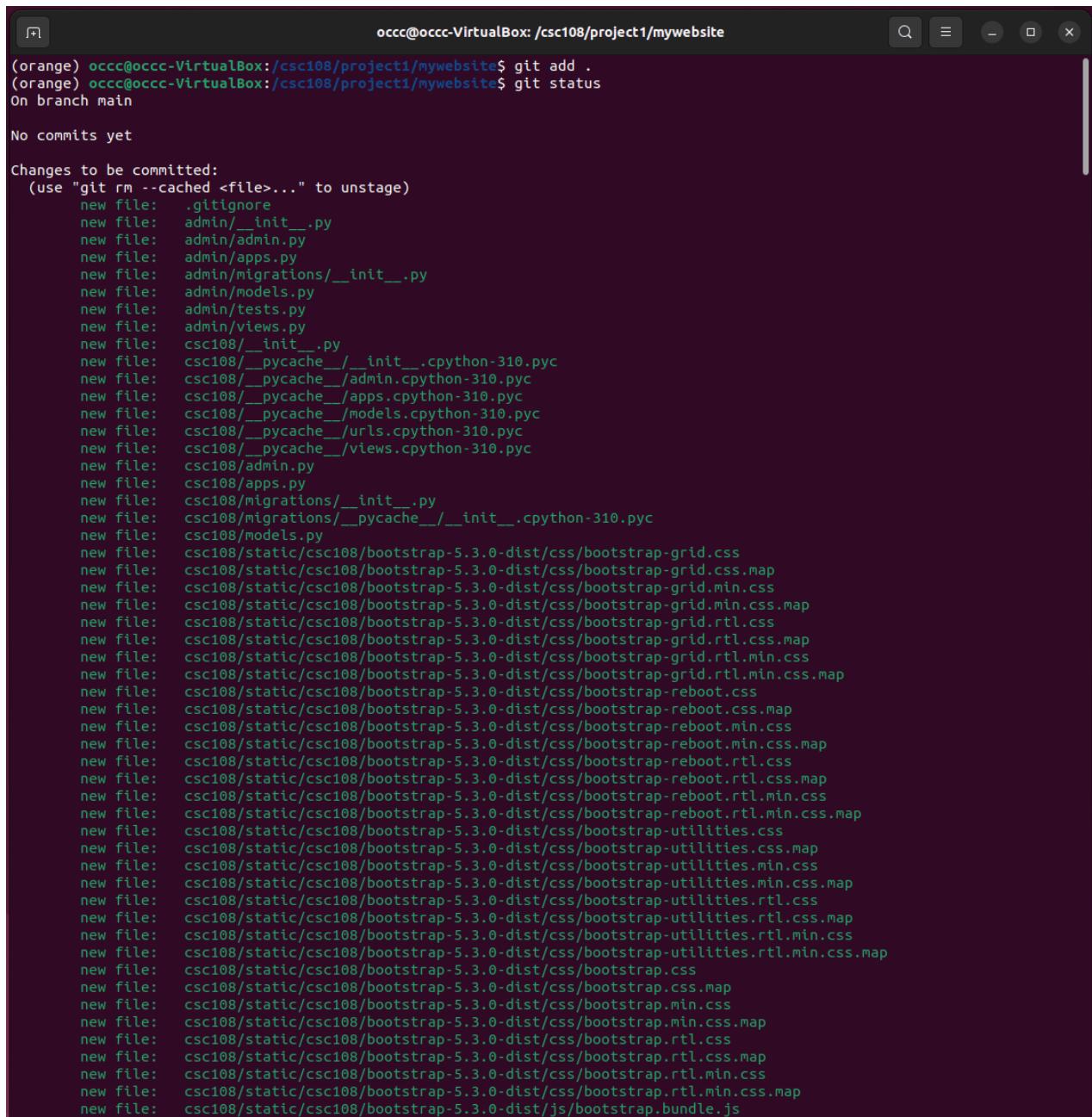
```
(orange) occc@occc-VirtualBox:/csc108/project1/mywebsite$ git status
On branch main

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
    .gitignore
    admin/
    csc108/
    faculty/
    manage.py
    mywebsite/
    students/
    testfile.txt
    users/

nothing added to commit but untracked files present (use "git add" to track)
(orange) occc@occc-VirtualBox:/csc108/project1/mywebsite$ █
```

add all our files to git



```
(orange) occc@occc-VirtualBox:/csc108/project1/mywebsite$ git add .
(orange) occc@occc-VirtualBox:/csc108/project1/mywebsite$ git status
On branch main

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
    new file:  .gitignore
    new file:  admin/__init__.py
    new file:  admin/admin.py
    new file:  admin/apps.py
    new file:  admin/migrations/__init__.py
    new file:  admin/models.py
    new file:  admin/tests.py
    new file:  admin/views.py
    new file:  csc108/__init__.py
    new file:  csc108/_pycache__/_init__.cpython-310.pyc
    new file:  csc108/_pycache__/_admin.cpython-310.pyc
    new file:  csc108/_pycache__/_apps.cpython-310.pyc
    new file:  csc108/_pycache__/_models.cpython-310.pyc
    new file:  csc108/_pycache__/_urls.cpython-310.pyc
    new file:  csc108/_pycache__/_views.cpython-310.pyc
    new file:  csc108/admin.py
    new file:  csc108/apps.py
    new file:  csc108/migrations/__init__.py
    new file:  csc108/models.py
    new file:  csc108/static/csc108/bootstrap-5.3.0-dist/css/bootstrap-grid.css
    new file:  csc108/static/csc108/bootstrap-5.3.0-dist/css/bootstrap-grid.css.map
    new file:  csc108/static/csc108/bootstrap-5.3.0-dist/css/bootstrap-grid.min.css
    new file:  csc108/static/csc108/bootstrap-5.3.0-dist/css/bootstrap-grid.min.css.map
    new file:  csc108/static/csc108/bootstrap-5.3.0-dist/css/bootstrap-grid rtl.css
    new file:  csc108/static/csc108/bootstrap-5.3.0-dist/css/bootstrap-grid rtl.css.map
    new file:  csc108/static/csc108/bootstrap-5.3.0-dist/css/bootstrap-grid rtl.min.css
    new file:  csc108/static/csc108/bootstrap-5.3.0-dist/css/bootstrap-grid rtl.min.css.map
    new file:  csc108/static/csc108/bootstrap-5.3.0-dist/css/bootstrap-reboot.css
    new file:  csc108/static/csc108/bootstrap-5.3.0-dist/css/bootstrap-reboot.css.map
    new file:  csc108/static/csc108/bootstrap-5.3.0-dist/css/bootstrap-reboot.min.css
    new file:  csc108/static/csc108/bootstrap-5.3.0-dist/css/bootstrap-reboot.min.css.map
    new file:  csc108/static/csc108/bootstrap-5.3.0-dist/css/bootstrap-reboot.rtl.css
    new file:  csc108/static/csc108/bootstrap-5.3.0-dist/css/bootstrap-reboot.rtl.css.map
    new file:  csc108/static/csc108/bootstrap-5.3.0-dist/css/bootstrap-reboot.rtl.min.css
    new file:  csc108/static/csc108/bootstrap-5.3.0-dist/css/bootstrap-reboot.rtl.min.css.map
    new file:  csc108/static/csc108/bootstrap-5.3.0-dist/css/bootstrap-utilities.css
    new file:  csc108/static/csc108/bootstrap-5.3.0-dist/css/bootstrap-utilities.css.map
    new file:  csc108/static/csc108/bootstrap-5.3.0-dist/css/bootstrap-utilities.min.css
    new file:  csc108/static/csc108/bootstrap-5.3.0-dist/css/bootstrap-utilities.min.css.map
    new file:  csc108/static/csc108/bootstrap-5.3.0-dist/css/bootstrap-utilities.rtl.css
    new file:  csc108/static/csc108/bootstrap-5.3.0-dist/css/bootstrap-utilities.rtl.css.map
    new file:  csc108/static/csc108/bootstrap-5.3.0-dist/css/bootstrap-utilities.rtl.min.css
    new file:  csc108/static/csc108/bootstrap-5.3.0-dist/css/bootstrap-utilities.rtl.min.css.map
    new file:  csc108/static/csc108/bootstrap-5.3.0-dist/css/bootstrap.css
    new file:  csc108/static/csc108/bootstrap-5.3.0-dist/css/bootstrap.css.map
    new file:  csc108/static/csc108/bootstrap-5.3.0-dist/css/bootstrap.min.css
    new file:  csc108/static/csc108/bootstrap-5.3.0-dist/css/bootstrap.min.css.map
    new file:  csc108/static/csc108/bootstrap-5.3.0-dist/css/bootstrap.rtl.css
    new file:  csc108/static/csc108/bootstrap-5.3.0-dist/css/bootstrap.rtl.css.map
    new file:  csc108/static/csc108/bootstrap-5.3.0-dist/css/bootstrap.rtl.min.css
    new file:  csc108/static/csc108/bootstrap-5.3.0-dist/css/bootstrap.rtl.min.css.map
    new file:  csc108/static/csc108/bootstrap-5.3.0-dist/js/bootstrap.bundle.js
```

git commit

The **git commit** command is one of the core primary functions of Git. Once you have staged the changes that you want to make in the Git repository using the "git add" command, you use the git "commit command" to actually record the changes in the repository's history.

The basic syntax for the "git commit" command is:

```
git commit -m "Your detailed commit message"
```

In this command, the **-m** option stands for "message", followed by a string that describes the changes made in the commit. This message should ideally be detailed and descriptive, so anyone who looks at the commit in the future will understand what changes were made.

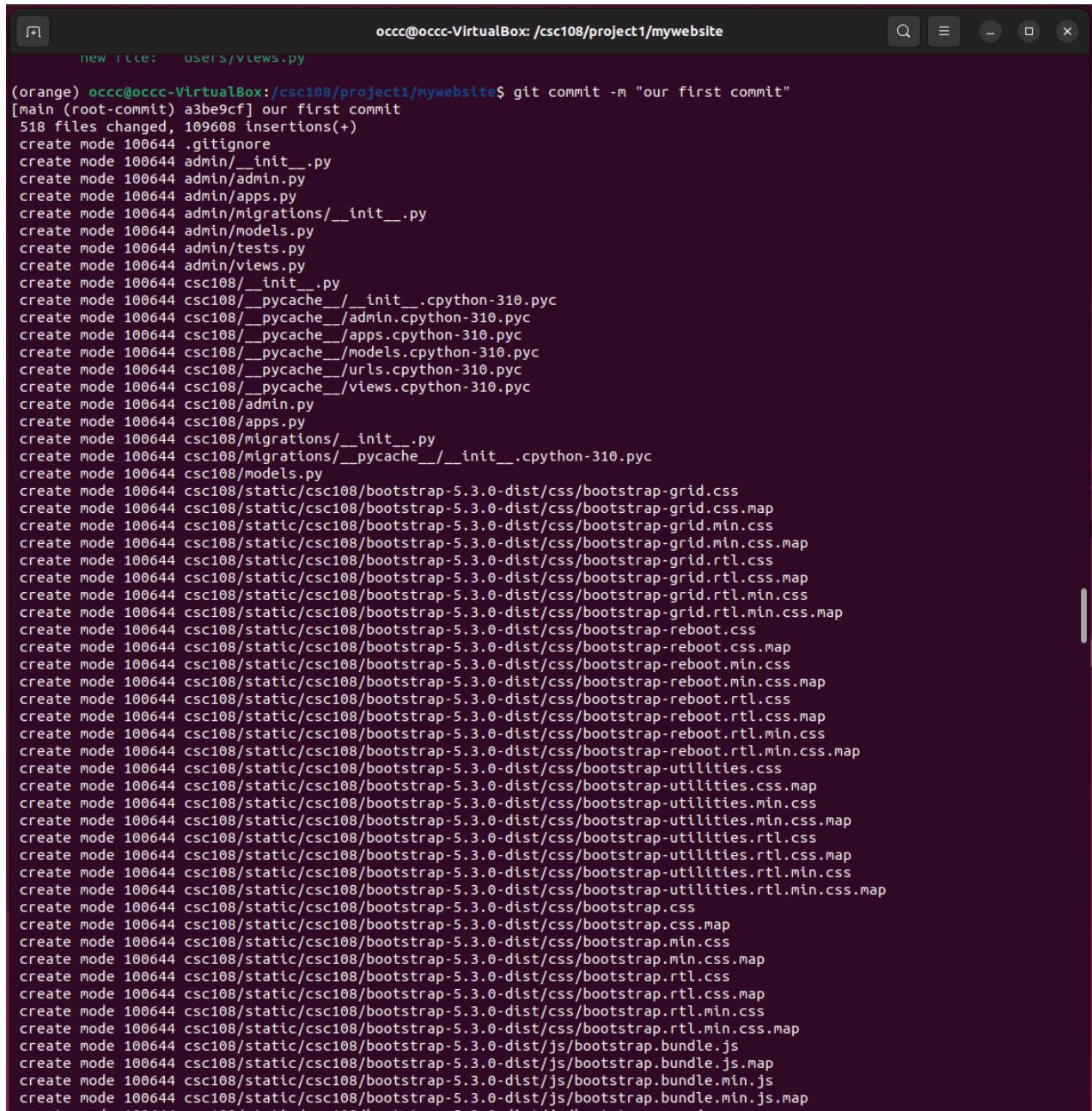
A commit in Git creates a stable "checkpoint" of your project, and it is also used for branching and merging. Each commit has a unique ID (a SHA-1 hash), which allows you to refer to specific commits in the repository history.

Once a commit is made, it can be pushed to a remote repository with "git push", or other collaborators can pull the commit to their local repositories with "git pull".

Please note, "git commit" only commits changes from the staging area (i.e., changes that have been added with "git add"). If you make further changes after "git add" and want them to be included in the commit, you will need to add those changes to the staging area again before committing. If you want to automatically stage all modified files (that are already tracked), skipping the git add step, you can use **git commit -a -m**.

The git commit command doesn't impact any remote repositories, it only saves changes locally. To share your commits with others, you need to use the "git push" command to send your commits to a remote repository.

make our first commit



```
(orange) occc@occc-VirtualBox:/csc108/project1/mywebsite$ git commit -m "our first commit"
[main (root-commit) a3be9cf] our first commit
 518 files changed, 109608 insertions(+)
 create mode 100644 .gitignore
 create mode 100644 admin/__init__.py
 create mode 100644 admin/admin.py
 create mode 100644 admin/apps.py
 create mode 100644 admin/migrations/__init__.py
 create mode 100644 admin/models.py
 create mode 100644 admin/tests.py
 create mode 100644 admin/views.py
 create mode 100644 csc108/__init__.py
 create mode 100644 csc108/_pycache__/__init__.cpython-310.pyc
 create mode 100644 csc108/_pycache__/_admin.cpython-310.pyc
 create mode 100644 csc108/_pycache__/_apps.cpython-310.pyc
 create mode 100644 csc108/_pycache__/_models.cpython-310.pyc
 create mode 100644 csc108/_pycache__/_urls.cpython-310.pyc
 create mode 100644 csc108/_pycache__/_views.cpython-310.pyc
 create mode 100644 csc108/admin.py
 create mode 100644 csc108/apps.py
 create mode 100644 csc108/migrations/__init__.py
 create mode 100644 csc108/migrations/_pycache__/_init__.cpython-310.pyc
 create mode 100644 csc108/models.py
 create mode 100644 csc108/static/csc108/bootstrap-5.3.0-dist/css/bootstrap-grid.css
 create mode 100644 csc108/static/csc108/bootstrap-5.3.0-dist/css/bootstrap-grid.css.map
 create mode 100644 csc108/static/csc108/bootstrap-5.3.0-dist/css/bootstrap-grid.min.css
 create mode 100644 csc108/static/csc108/bootstrap-5.3.0-dist/css/bootstrap-grid.min.css.map
 create mode 100644 csc108/static/csc108/bootstrap-5.3.0-dist/css/bootstrap-grid rtl.css
 create mode 100644 csc108/static/csc108/bootstrap-5.3.0-dist/css/bootstrap-grid rtl.css.map
 create mode 100644 csc108/static/csc108/bootstrap-5.3.0-dist/css/bootstrap-grid rtl.min.css
 create mode 100644 csc108/static/csc108/bootstrap-5.3.0-dist/css/bootstrap-grid rtl.min.css.map
 create mode 100644 csc108/static/csc108/bootstrap-reboot.css
 create mode 100644 csc108/static/csc108/bootstrap-reboot.css.map
 create mode 100644 csc108/static/csc108/bootstrap-reboot.min.css
 create mode 100644 csc108/static/csc108/bootstrap-reboot.min.css.map
 create mode 100644 csc108/static/csc108/bootstrap-reboot.rtl.css
 create mode 100644 csc108/static/csc108/bootstrap-reboot.rtl.css.map
 create mode 100644 csc108/static/csc108/bootstrap-reboot.rtl.min.css
 create mode 100644 csc108/static/csc108/bootstrap-reboot.rtl.min.css.map
 create mode 100644 csc108/static/csc108/bootstrap-utilities.css
 create mode 100644 csc108/static/csc108/bootstrap-utilities.css.map
 create mode 100644 csc108/static/csc108/bootstrap-utilities.min.css
 create mode 100644 csc108/static/csc108/bootstrap-utilities.min.css.map
 create mode 100644 csc108/static/csc108/bootstrap-utilities rtl.css
 create mode 100644 csc108/static/csc108/bootstrap-utilities rtl.css.map
 create mode 100644 csc108/static/csc108/bootstrap-utilities rtl.min.css
 create mode 100644 csc108/static/csc108/bootstrap-utilities rtl.min.css.map
 create mode 100644 csc108/static/csc108/bootstrap.css
 create mode 100644 csc108/static/csc108/bootstrap.css.map
 create mode 100644 csc108/static/csc108/bootstrap.min.css
 create mode 100644 csc108/static/csc108/bootstrap.min.css.map
 create mode 100644 csc108/static/csc108/bootstrap rtl.css
 create mode 100644 csc108/static/csc108/bootstrap rtl.css.map
 create mode 100644 csc108/static/csc108/bootstrap rtl.min.css
 create mode 100644 csc108/static/csc108/bootstrap rtl.min.css.map
 create mode 100644 csc108/static/csc108/bootstrap.bundle.js
 create mode 100644 csc108/static/csc108/bootstrap-5.3.0-dist/js/bootstrap.bundle.js.map
 create mode 100644 csc108/static/csc108/bootstrap-5.3.0-dist/js/bootstrap.bundle.min.js
 create mode 100644 csc108/static/csc108/bootstrap-5.3.0-dist/js/bootstrap.bundle.min.js.map
 create mode 100644 csc108/static/csc108/bootstrap-5.3.0-dist/js/bootstrap.bundle.min.js.map
```

```
(orange) occc@occc-VirtualBox:/csc108/project1/mywebsite$ git status
On branch main
nothing to commit, working tree clean
(orange) occc@occc-VirtualBox:/csc108/project1/mywebsite$
```

make change to a file

```
(orange) occc@occc-VirtualBox:/csc108/project1/mywebsite$ ls
admin csc108 db.sqlite3 faculty manage.py mywebsite students testfile.txt users
(orange) occc@occc-VirtualBox:/csc108/project1/mywebsite$ more testfile.txt
this is a test
(orange) occc@occc-VirtualBox:/csc108/project1/mywebsite$ echo "Orange County Community College" > testfile.txt
(orange) occc@occc-VirtualBox:/csc108/project1/mywebsite$ more testfile.txt
Orange County Community College
(orange) occc@occc-VirtualBox:/csc108/project1/mywebsite$
```

Git Status after a change in the file.

```
(orange) occc@occc-VirtualBox:/csc108/project1/mywebsite$ git status
On branch main
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
    modified:   testfile.txt

no changes added to commit (use "git add" and/or "git commit -a")
(orange) occc@occc-VirtualBox:/csc108/project1/mywebsite$
```

git diff

The git diff command is a multi-use command in Git that, when executed, shows differences between commits, commit and working tree, etc. It essentially shows you what has been changed but not yet staged or committed.

git diff

When run without any options, git diff will show you any uncommitted changes you've made since your last commit, specifically changes that you haven't staged yet. These changes could be new files you've added (but not staged), modifications to existing files, or deletions.

Some option to the diff command:

- **git diff –staged or git diff –cached:** These commands will show you the differences between the staging area and the last commit. This is useful to review before making a commit.
- **git diff HEAD:** This will show all changes in the working tree since your last commit; in other words, it includes both staged and unstaged changes.
- **git diff <commit1> <commit2>:** This will show the differences between two specific commits.
- **git diff <branch1>..<branch2>:** This will show the differences between two branches.

In the output of the git diff command, lines that are added are shown in green with a plus sign (+) at the start of the line, and lines that are removed are shown in red with a minus sign (-) at the start of the line.

It's important to note that git diff only shows the differences and doesn't actually change any files in your repository. It's a tool for inspecting changes and doesn't alter the state of the repository.

```
(orange) occc@occc-VirtualBox:/csc108/project1/mywebsite$ git diff
diff --git a/testfile.txt b/testfile.txt
index 90bfcb5..810c5b7 100644
--- a/testfile.txt
+++ b/testfile.txt
@@ -1 +1 @@
-this is a test
+Orange County Community College
(orange) occc@occc-VirtualBox:/csc108/project1/mywebsite$
```

add changed file to staging again

```
(orange) occc@occc-VirtualBox:/csc108/project1/mywebsite$ git status
On branch main
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
    modified:   testfile.txt

no changes added to commit (use "git add" and/or "git commit -a")
(orange) occc@occc-VirtualBox:/csc108/project1/mywebsite$ git add testfile.txt
(orange) occc@occc-VirtualBox:/csc108/project1/mywebsite$ git status
On branch main
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    modified:   testfile.txt

(orange) occc@occc-VirtualBox:/csc108/project1/mywebsite$
```

bypass staging and commit changed files

git commit -a -m

```
(orange) occc@occc-VirtualBox:/csc108/project1/mywebsite$ git status
On branch main
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
    modified:   testfile.txt

no changes added to commit (use "git add" and/or "git commit -a")
(orange) occc@occc-VirtualBox:/csc108/project1/mywebsite$ git commit -a -m "changed testfile.txt"
[main fd2ecbc] changed testfile.txt
  1 file changed, 1 insertion(+), 1 deletion(-)
(orange) occc@occc-VirtualBox:/csc108/project1/mywebsite$
```

restore files

If we delete the testfile.txt from our system we can restore it back from git.

git restore

The **git restore** command is used to revert changes that you've made in your working directory but haven't yet committed. It's a way of undoing local changes and restoring files to a previous state.

There are a couple of key ways you can use git restore:

- **Unstage files from the staging area:** If you've added files to the staging area with git add but haven't committed them yet, and you decide you don't want to commit them, you can use git restore to remove them from the staging area.

```
git restore --staged myfile.txt
```

This command will unstage myfile.txt. The changes you've made to the file won't be lost, but the file will no longer be in the staging area ready for commit.

- **Discard changes in the working directory:** If you've made changes to a file but decide you want to discard them and return the file to the state it was in at the last commit, you can use git restore

```
git restore myfile.txt
```

This command will restore myfile.txt to the state it was in at the last commit, discarding any changes you've made since then.

Please note that "git restore" is a destructive command, and any changes it undoes can't be recovered. So, you should be sure you really want to discard changes before running it.

```
(orange) occc@occc-VirtualBox:/csc108/project1/mywebsite$ ls
admin csc108 db.sqlite3 faculty manage.py mywebsite students testfile.txt users
(orange) occc@occc-VirtualBox:/csc108/project1/mywebsite$ rm testfile.txt
(orange) occc@occc-VirtualBox:/csc108/project1/mywebsite$ ls
admin csc108 db.sqlite3 faculty manage.py mywebsite students users
(orange) occc@occc-VirtualBox:/csc108/project1/mywebsite$ git status
On branch main
Changes not staged for commit:
  (use "git add/rm <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
    deleted:    testfile.txt

no changes added to commit (use "git add" and/or "git commit -a")
(orange) occc@occc-VirtualBox:/csc108/project1/mywebsite$ git restore testfile.txt
(orange) occc@occc-VirtualBox:/csc108/project1/mywebsite$ ls
admin csc108 db.sqlite3 faculty manage.py mywebsite students testfile.txt users
(orange) occc@occc-VirtualBox:/csc108/project1/mywebsite$ git status
On branch main
nothing to commit, working tree clean
(orange) occc@occc-VirtualBox:/csc108/project1/mywebsite$ more testfile.txt
Orange County Community College
(orange) occc@occc-VirtualBox:/csc108/project1/mywebsite$ █
```

git mv

The `git mv` command is used in Git to move or rename a file or directory in your repository. It's essentially a convenience command that performs several steps in one go.

syntax for the "git mv" command is:

```
git mv <old-filename> <new-filename>
```

This command does three things:

- Renames the file in your working directory.
- Removes the old file from the staging area.
- Adds the new file to the staging area.

to rename a file called `oldname.txt` to `newname.txt`, you'd use the following command:

```
git mv oldname.txt newname.txt
```

Once you've run the `git mv` command, you still need to commit the change to record it in your repository's history. Until you do this, the renaming/moving is not "official". So you would then run:

```
git commit -m "Renamed oldname.txt to newname.txt"
```

If you want to move a file into a different directory, you can do that with `git mv` as well:

```
git mv myfile.txt mydirectory/
```

This command will move `myfile.txt` into the `mydirectory/` directory. Again, you'd need to commit the change to record it.

While `git mv` is a convenient shorthand, you could perform the same actions with regular Unix commands (or their equivalent on your operating system) and other git commands. For instance, you could use `mv oldname.txt newname.txt` to rename a file, then use `git add newname.txt` to stage the new file and `git rm oldname.txt` to remove the old one.

Lets test it out by renaming our `testfile.txt` to `occc.txt`

```
(orange) occc@occc-VirtualBox:/csc108/project1/mywebsite$ git mv "testfile.txt" "occc.txt"
(orange) occc@occc-VirtualBox:/csc108/project1/mywebsite$ ls
admin csc108 db.sqlite3 faculty manage.py mywebsite occc.txt students users
(orange) occc@occc-VirtualBox:/csc108/project1/mywebsite$ git status
On branch main
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    renamed:   testfile.txt -> occc.txt

(orange) occc@occc-VirtualBox:/csc108/project1/mywebsite$ git commit -m "Renammed testfile.txt"
[main 4477f36] Renammed testfile.txt to occc.txt
 1 file changed, 0 insertions(+), 0 deletions(-)
 rename testfile.txt => occc.txt (100%)
(orange) occc@occc-VirtualBox:/csc108/project1/mywebsite$
```

git log

The `git log` command is used in Git to display committed snapshots. It lets you list the project history, showing the commit hash, the author, the date, and the commit message for each commit. The output is displayed in reverse chronological order (the most recent commit is shown first).

Syntax:

```
git log
```

```
(orange) occc@occc-VirtualBox:/csc108/project1/mywebsite$ git log
commit 4477f36601eb52b9d6bc5cf3ca7dbce7093c1c5d (HEAD -> main)
Author: miroslav Krajca <miroslavkrajca@sunnyorange.edu>
Date:   Wed Jun 28 14:31:28 2023 -0400

  Renammed testfile.txt to occc.txt

commit fd2ecbc313363a802dbd1ed1da2150697fac1baf
Author: miroslav Krajca <miroslavkrajca@sunnyorange.edu>
Date:   Wed Jun 28 14:06:40 2023 -0400

  changed testfile.txt

commit a3be9cfef4c4e21e1e88bce892c18e2f647e3a7
Author: miroslav Krajca <miroslavkrajca@sunnyorange.edu>
Date:   Wed Jun 28 12:41:55 2023 -0400

  our first commit
(orange) occc@occc-VirtualBox:/csc108/project1/mywebsite$ █
```

There are several options that you can use with `git log` to adjust its output:

- **git log -n <limit>:** Limit the number of commits by <limit>. For example, git log -n 3 will show only the last three commits.
- **git log --oneline:** This will shorten the output to one line per commit, showing just the commit hash and the commit message.
- **git log --graph:** This will display an ASCII graph of the branch and merge history beside the log output.
- **git log --author="":** This will limit the logs to the ones with the specified author. Replace <pattern> with the author's name.
- **git log --grep="":** This will search for <pattern> in the commit message.
- **git log <file>:** This will show changes over time for the specific file.
- **git log --since=<time>:** This shows commits made since a certain date. Replace <time> with a date or period such as 2.weeks.

There are many more options and filters you can use to customize the command's output.

amending a commit

To amend a Git commit, you can use the **--amend** option with the git commit command. This allows you to modify the most recent commit. You might want to do this if you made a mistake in your commit message, forgot to add some files, or made a typo in your code that you didn't notice until after you committed.

- **Modifying the commit message:** If you just want to modify the commit message of the most recent commit, you can use the --amend option like this:

```
git commit --amend -m "New commit message"
```

When you run this command, the commit message of the most recent commit will be changed to "New commit message".

- **Adding forgotten files to the commit:** If you forgot to add some files to your commit, you can do this:

- **Stage the forgotten files with git add:**

```
git add forgotten-file.txt
```

- **Then amend the commit and optionally update the commit message:**

```
git commit --amend -m "New commit message"
```

When you run these commands, forgotten-file.txt will be included in the most recent commit, and the commit message of the most recent commit will be changed to "New commit message".

Please note that when you amend a commit, it actually creates a new commit with a new SHA hash and replaces the old commit. This is important to understand if you've already pushed your commits to a remote repository, because you'll need to force push the amended commit with `git push --force`. Be aware that force pushing can be dangerous if others are working on the same branch, as it can overwrite their changes.

```
(orange) occc@occc-VirtualBox:/csc108/project1/mywebsite$ git commit -m "Renamed testfile.txt to occc.txt" --amend
[main 780bb8c] Renamed testfile.txt to occc.txt
  Date: Wed Jun 28 14:31:28 2023 -0400
  1 file changed, 0 insertions(+), 0 deletions(-)
  rename testfile.txt => occc.txt (100%)
(orange) occc@occc-VirtualBox:/csc108/project1/mywebsite$ git status
On branch main
nothing to commit, working tree clean
(orange) occc@occc-VirtualBox:/csc108/project1/mywebsite$ git log
commit 780bb8cc6ad837477e15d23efdf6cbc283c5063e (HEAD -> main)
Author: miroslav Krajca <miroslavkrajca@sunnyorange.edu>
Date:   Wed Jun 28 14:31:28 2023 -0400

    Renamed testfile.txt to occc.txt

commit fd2ecbc313363a802dbd1ed1da2150697fac1baf
Author: miroslav Krajca <miroslavkrajca@sunnyorange.edu>
Date:   Wed Jun 28 14:06:40 2023 -0400

    changed testfile.txt

commit a3be9cfef4c4e21e1e88bce892c18e2f647e3a7
Author: miroslav Krajca <miroslavkrajca@sunnyorange.edu>
Date:   Wed Jun 28 12:41:55 2023 -0400

    our first commit
(orange) occc@occc-VirtualBox:/csc108/project1/mywebsite$
```

code branches

In Git, a branch is essentially a pointer to a specific commit. It represents an independent line of development in a project, where you can create, edit, and delete files separately without affecting other branches.

The main use of branches is to isolate work when you're developing new features or fixes, so they're not mixed up with your main line of development. When you're done with the development work on the branch, you can merge it back into your main line of development (the 'main' branch).

To create a new branch in Git, you use the **git branch** command followed by the name of the new branch.

git branch

```
(orange) occc@occc-VirtualBox:/csc108/project1/mywebsite$ git branch
* main
(orange) occc@occc-VirtualBox:/csc108/project1/mywebsite$ git branch orange
(orange) occc@occc-VirtualBox:/csc108/project1/mywebsite$ git branch
* main
  orange
(orange) occc@occc-VirtualBox:/csc108/project1/mywebsite$
```

The ***** indicates which branch is the active branch.

switch branch

```
(orange) occc@occc-VirtualBox:/csc108/project1/mywebsite$ git branch
* main
  orange
(orange) occc@occc-VirtualBox:/csc108/project1/mywebsite$ git switch orange
Switched to branch 'orange'
(orange) occc@occc-VirtualBox:/csc108/project1/mywebsite$ git branch
  main
* orange
(orange) occc@occc-VirtualBox:/csc108/project1/mywebsite$ █
```

Edit our occc.txt file and commit it to the new branch.

```
(orange) occc@occc-VirtualBox:/csc108/project1/mywebsite$ git branch
  main
* orange
(orange) occc@occc-VirtualBox:/csc108/project1/mywebsite$ git status
On branch orange
nothing to commit, working tree clean
(orange) occc@occc-VirtualBox:/csc108/project1/mywebsite$ more occc.txt
Orange County Community College
(orange) occc@occc-VirtualBox:/csc108/project1/mywebsite$ echo "occc1234" > occc.txt
(orange) occc@occc-VirtualBox:/csc108/project1/mywebsite$ more occc.txt
occc1234
(orange) occc@occc-VirtualBox:/csc108/project1/mywebsite$ git status
On branch orange
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
    (use "git restore <file>..." to discard changes in working directory)
      modified:   occc.txt

no changes added to commit (use "git add" and/or "git commit -a")
(orange) occc@occc-VirtualBox:/csc108/project1/mywebsite$ git commit -a -m "change occc."
[orange 0ee6232] change occc.txt in orange branch
 1 file changed, 1 insertion(+), 1 deletion(-)
(orange) occc@occc-VirtualBox:/csc108/project1/mywebsite$ git status
On branch orange
nothing to commit, working tree clean
(orange) occc@occc-VirtualBox:/csc108/project1/mywebsite$ █
```

switch branch back to main

```
(orange) occc@occc-VirtualBox:/csc108/project1/mywebsite$ git switch main
Switched to branch 'main'
(orange) occc@occc-VirtualBox:/csc108/project1/mywebsite$ git branch
* main
  orange
(orange) occc@occc-VirtualBox:/csc108/project1/mywebsite$ more occc.txt
Orange County Community College
(orange) occc@occc-VirtualBox:/csc108/project1/mywebsite$ █
```

When we switched back to the main branch the changes that we did to the occc.txt file are no longer there. Switching branches also applies any and all committed changes to the code.

switch branch to orange

```
(orange) occc@occc-VirtualBox:/csc108/project1/mywebsite$ git branch
* main
  orange
(orange) occc@occc-VirtualBox:/csc108/project1/mywebsite$ more occc.txt
Orange County Community College
(orange) occc@occc-VirtualBox:/csc108/project1/mywebsite$ git switch orange
Switched to branch 'orange'
(orange) occc@occc-VirtualBox:/csc108/project1/mywebsite$ more occc.txt
occc1234
(orange) occc@occc-VirtualBox:/csc108/project1/mywebsite$ █
```

git merging

The **git merge** command is used to combine two branches in Git. It takes the contents of a source branch and integrates it with the target branch.

Simple example:

- You're currently on your main branch.
- You create a new branch called feature_branch, make some changes there, and commit them.
- After completing your changes on the feature_branch, you switch back to the main branch by using git switch main.
- Then, you use the git merge command to merge the feature_branch into main:

```
git merge feature_branch
```

- If there are no conflicts between the branches, Git will perform a "fast-forward merge", moving the main branch pointer to point to the same commit as feature_branch.
- If there are conflicting changes on the two branches, Git will attempt to auto-merge them. If Git can't figure out how to auto-merge, it will create a merge conflict that you have to resolve manually.

In the case of a merge conflict, the conflicted files will include special markers that indicate the changes from each branch. You'll need to edit the files to resolve the conflicts, and then you can commit the resolved files.

The git merge command is a critical part of collaborating with Git. It allows multiple developers to work on different features in parallel, and then merge their changes back together in an organized and controlled manner.

Note: Merging is not the only way to integrate changes from one branch into another in Git. You can also use the git rebase command, which applies your changes on top of the changes from another branch. The choice between merging and rebasing depends on the specifics of your project and workflow.

switch branch back to main

```
(orange) occc@occc-VirtualBox:/csc108/project1/mywebsite$ git switch main
Switched to branch 'main'
(orange) occc@occc-VirtualBox:/csc108/project1/mywebsite$ git merge orange -m "merge wi
Updating 780bb8c..0ee6232
Fast-forward (no commit created; -m option ignored)
  occc.txt | 2 ++
  1 file changed, 1 insertion(+), 1 deletion(-)
(orange) occc@occc-VirtualBox:/csc108/project1/mywebsite$ more occc.txt
occc1234
(orange) occc@occc-VirtualBox:/csc108/project1/mywebsite$ more status
more: cannot open status: No such file or directory
(orange) occc@occc-VirtualBox:/csc108/project1/mywebsite$ git status
On branch main
nothing to commit, working tree clean
(orange) occc@occc-VirtualBox:/csc108/project1/mywebsite$
```

git branch -d (delete a branch)

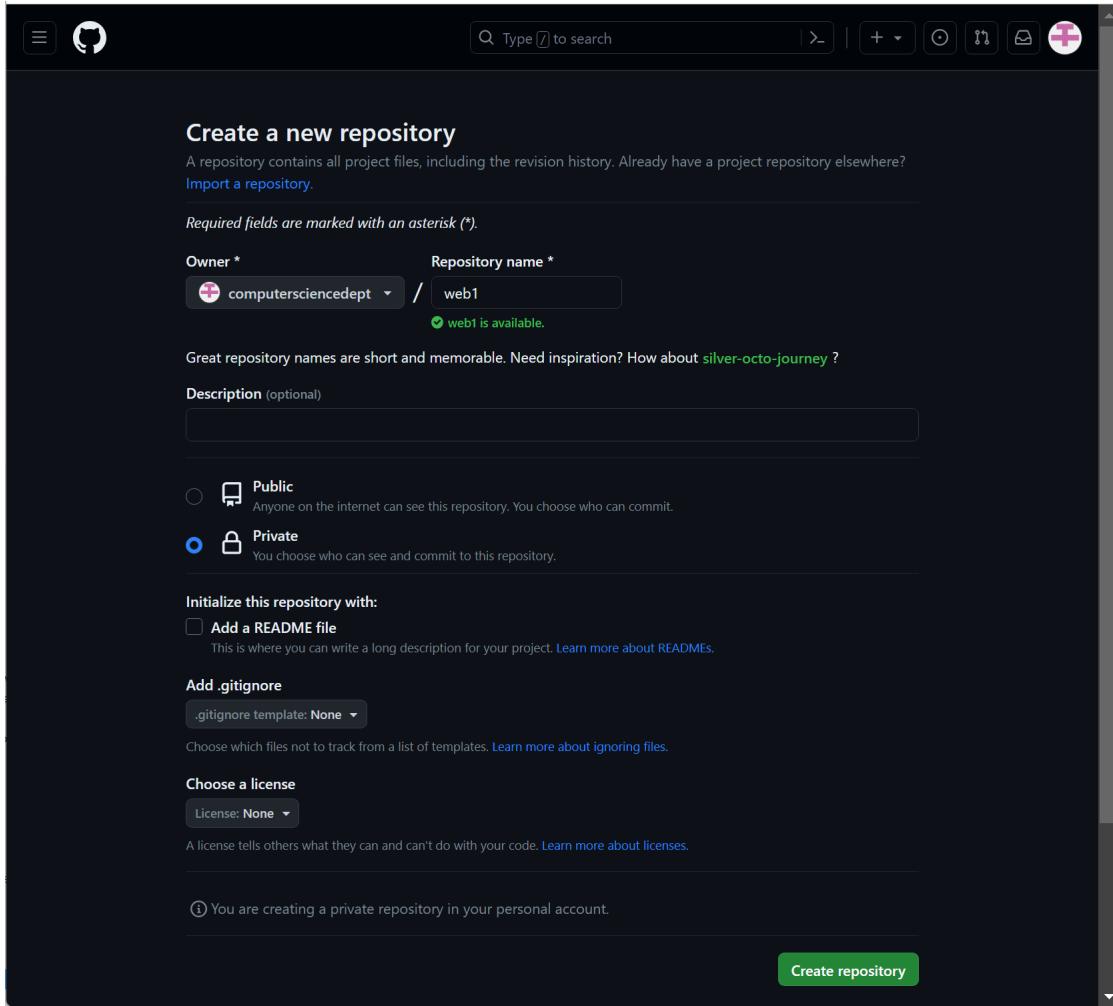
```
(orange) occc@occc-VirtualBox:/csc108/project1/mywebsite$ git branch
* main
  orange
(orange) occc@occc-VirtualBox:/csc108/project1/mywebsite$ git branch -d orange
Deleted branch orange (was 0ee6232).
(orange) occc@occc-VirtualBox:/csc108/project1/mywebsite$ git branch
* main
(orange) occc@occc-VirtualBox:/csc108/project1/mywebsite$
```

9.8 ➤ github

Create a github account. There are free accounts as well as education accounts for students and teachers.

https://education.github.com/discount_requests/application

9.8.1 ➤ create a repository



Create a repository.

Github will provide some basic information on how to set up a git and push it to to github.

Quick setup — if you've done this kind of thing before

Set up in Desktop or HTTPS SSH git@github.com:computersciencedept/web1.git

Get started by creating a new file or uploading an existing file. We recommend every repository include a `README`, `LICENSE`, and `.gitignore`.

...or create a new repository on the command line

```
echo "# web1" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin git@github.com:computersciencedept/web1.git
git push -u origin main
```

...or push an existing repository from the command line

```
git remote add origin git@github.com:computersciencedept/web1.git
git branch -M main
git push -u origin main
```

...or import code from another repository

You can initialize this repository with code from a Subversion, Mercurial, or TFS project.

[Import code](#)

The above instructions are for ssh if you click the https the instructions will change to https.

git branch -M main

This will rename your branch name to "main"

If we follow the instructions for ssh(preferred way to use git) they will not work.

```
occc@occc-VirtualBox:/csc108/project1/mywebsite$ git remote add origin git@github.com:computersciencedept/web1.git
occc@occc-VirtualBox:/csc108/project1/mywebsite$ git push -u origin main
The authenticity of host 'github.com (140.82.114.4)' can't be established.
ED25519 key fingerprint is SHA256:+DiY3wvvV6TuJJhbZisF/zLDA0zPMSvHdkr4UvC0qU.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'github.com' (ED25519) to the list of known hosts.
git@github.com: Permission denied (publickey).
fatal: Could not read from remote repository.

Please make sure you have the correct access rights
and the repository exists.
occc@occc-VirtualBox:/csc108/project1/mywebsite$
```

This has to do with ssh keys not existing or not imported to github. Essentially github says that they cannot verify who you are and will not give permission to upload your code until a verification can be made.

Error: Permission denied (publickey)

ssh -vT git@github.com

```
(orange) occc@occc-VirtualBox:/csc108/project1/mywebsite$ ssh -vT git@github.com
OpenSSH_8.9p1 Ubuntu-3ubuntu0.1, OpenSSL 3.0.2 15 Mar 2022
debug1: Reading configuration data /etc/ssh/ssh_config
debug1: /etc/ssh/ssh_config line 19: include /etc/ssh/ssh_config.d/*.conf matched no files
debug1: /etc/ssh/ssh_config line 21: Applying options for *
debug1: Connecting to github.com [140.82.112.3] port 22.
debug1: Connection established.
debug1: identity file /home/occc/.ssh/id_rsa type -1
debug1: identity file /home/occc/.ssh/id_rsa-cert type -1
debug1: identity file /home/occc/.ssh/id_ecdsa type -1
debug1: identity file /home/occc/.ssh/id_ecdsa-cert type -1
debug1: identity file /home/occc/.ssh/id_ecdsa_sk type -1
debug1: identity file /home/occc/.ssh/id_ecdsa_sk-cert type -1
debug1: identity file /home/occc/.ssh/id_ed25519 type -1
debug1: identity file /home/occc/.ssh/id_ed25519-cert type -1
debug1: identity file /home/occc/.ssh/id_ed25519_sk type -1
debug1: identity file /home/occc/.ssh/id_ed25519_sk-cert type -1
debug1: identity file /home/occc/.ssh/id_xmss type -1
debug1: identity file /home/occc/.ssh/id_xmss-cert type -1
debug1: identity file /home/occc/.ssh/id_dsa type -1
debug1: identity file /home/occc/.ssh/id_dsa-cert type -1
debug1: Local version string SSH-2.0-OpenSSH_8.9p1 Ubuntu-3ubuntu0.1
debug1: Remote protocol version 2.0, remote software version babeld-2ff5bbdf
debug1: compat_banner: no match: babeld-2ff5bbdf
debug1: Authenticating to github.com:22 as 'git'
debug1: load_hostkeys: fopen /home/occc/.ssh/known_hosts2: No such file or directory
debug1: load_hostkeys: fopen /etc/ssh/ssh_known_hosts: No such file or directory
debug1: load_hostkeys: fopen /etc/ssh/ssh_known_hosts2: No such file or directory
debug1: SSH2_MSG_KEXINIT sent
debug1: SSH2_MSG_KEXINIT received
debug1: kex: algorithm: curve25519-sha256
debug1: kex: host key algorithm: ssh-ed25519
debug1: kex: server->client cipher: chacha20-poly1305@openssh.com MAC: <implicit> compression: none
debug1: kex: client->server cipher: chacha20-poly1305@openssh.com MAC: <implicit> compression: none
debug1: expecting SSH2_MSG_KEX_ECDH_REPLY
debug1: SSH2_MSG_KEX_ECDH_REPLY received
debug1: Server host key: ssh-ed25519 SHA256:+DiY3WvvV6TuJJhbpbZisF/zLDA0zPMsvHdkr4UvC0QU
debug1: load_hostkeys: fopen /home/occc/.ssh/known_hosts2: No such file or directory
debug1: load_hostkeys: fopen /etc/ssh/ssh_known_hosts: No such file or directory
debug1: load_hostkeys: fopen /etc/ssh/ssh_known_hosts2: No such file or directory
debug1: Host 'github.com' is known and matches the ED25519 host key.
debug1: Found key in /home/occc/.ssh/known_hosts1
debug1: rekey out after 134217728 blocks
debug1: SSH2_MSG_NEWKEYS sent
debug1: expecting SSH2_MSG_NEWKEYS
debug1: SSH2_MSG_NEWKEYS received
debug1: rekey in after 134217728 blocks
debug1: get_agent_identities: bound agent to hostkey
debug1: get_agent_identities: ssh_fetch_identitylist: agent contains no identities
debug1: Will attempt key: /home/occc/.ssh/id_rsa
debug1: Will attempt key: /home/occc/.ssh/id_ecdsa
debug1: Will attempt key: /home/occc/.ssh/id_ecdsa_sk
debug1: Will attempt key: /home/occc/.ssh/id_ed25519
debug1: Will attempt key: /home/occc/.ssh/id_ed25519_sk
debug1: Will attempt key: /home/occc/.ssh/id_xmss
debug1: Will attempt key: /home/occc/.ssh/id_dsa
debug1: SSH2_MSG_EXT_INFO received
debug1: kex_input_ext_info: server-sig-algs=<ssh-ed25519-cert-v01@openssh.com,ecdsa-sha2-nistp521-cert-v01@openssh.com,ecdsa-sha2-nistp384-cert-v01@openssh.com,ecdsa-sha2-nistp256-cert-v01@openssh.com,sk-ssh-ed25519-cert-v01@openssh.com,sk-ecdsa-sha2-nistp256-cert-v01@openssh.com,rsa-sha2-512-cert-v01@openssh.com,rsa-sha2-256-cert-v01@openssh.com,ssh-rsa-cert-v01@openssh.com,sk-ssh-ed25519@openssh.com,sk-ecdsa-sha2-nistp256@openssh.com,ssh-ed25519,ecdsa-sha2-nistp521,ecdsa-sha2-nistp384,ecdsa-sha2-nistp256,rsa-sha2-512,rsa-sha2-256,ssh-rsa>
debug1: SSH2_MSG_SERVICE_ACCEPT received
debug1: Authentications that can continue: publickey
debug1: Next authentication method: publickey
debug1: Trying private key: /home/occc/.ssh/id_rsa
debug1: Trying private key: /home/occc/.ssh/id_ecdsa
debug1: Trying private key: /home/occc/.ssh/id_ecdsa_sk
debug1: Trying private key: /home/occc/.ssh/id_ed25519
debug1: Trying private key: /home/occc/.ssh/id_ed25519_sk
debug1: Trying private key: /home/occc/.ssh/id_xmss
debug1: Trying private key: /home/occc/.ssh/id_dsa
debug1: No more authentication methods to try.
git@github.com: Permission denied (publickey).
(orange) occc@occc-VirtualBox:/csc108/project1/mywebsite$
```

We get the permission denied(publickey) error.

9.8.2 verify or create ssh keys

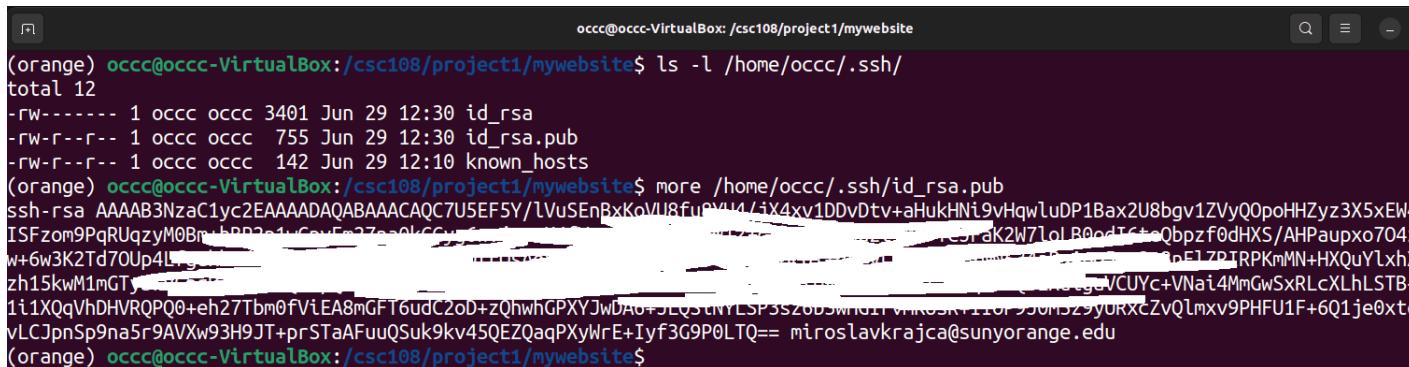
```
(orange) occc@occc-VirtualBox:/csc108/project1/mywebsite$ ssh -T git@github.com
git@github.com: Permission denied (publickey).
(orange) occc@occc-VirtualBox:/csc108/project1/mywebsite$ ls -l /home/occc/.ssh/
total 4
-rw-r--r-- 1 occc occc 142 Jun 29 12:10 known_hosts
(orange) occc@occc-VirtualBox:/csc108/project1/mywebsite$ 
```

ssh-keygen -t rsa -b 4096 -C "<your email addr>"

```
Generating public/private rsa key pair.
Enter file in which to save the key (/home/occc/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/occc/.ssh/id_rsa
Your public key has been saved in /home/occc/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:l/71STtTPy5G6bMHPok9J0zuHNM1NuTOXD7zjp+uU miroslavkrajca@sunnyorange.edu
The key's randomart image is:
+---[RSA 4096]---+
| |
| |
| . . o.|
| S +. o.*|
| .... .00|
| ..= @@|
| XoXOO|
| .=*XE|
+---[SHA256]---+
(orange) occc@occc-VirtualBox:/csc108/project1/mywebsite$ 
```

DO NOT USE ANY PASSPHRASE!!!

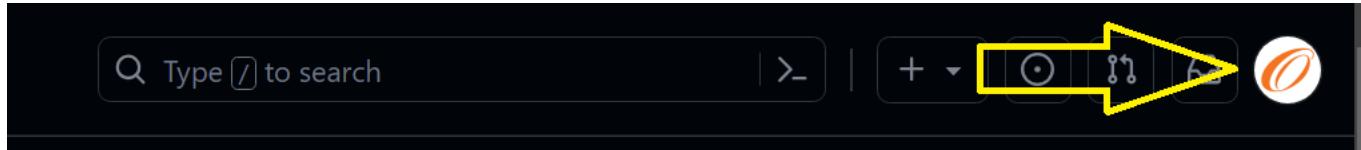
ssh public key



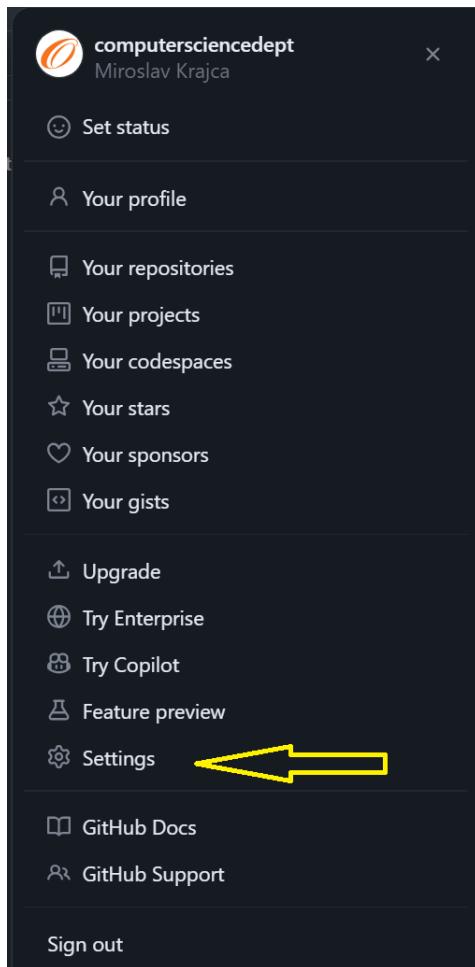
```
occc@occc-VirtualBox:/csc108/project1/mywebsite$ ls -l /home/occc/.ssh/
total 12
-rw----- 1 occc occc 3401 Jun 29 12:30 id_rsa
-rw-r--r-- 1 occc occc 755 Jun 29 12:30 id_rsa.pub
-rw-r--r-- 1 occc occc 142 Jun 29 12:10 known_hosts
(orange) occc@occc-VirtualBox:/csc108/project1/mywebsite$ more /home/occc/.ssh/id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAQABAAQAC7U5EF5Y/lVuSEnBxKoVJ8fuPv4M/jx4xv1DDvDtv+aHukHNi9vHqwluDp1Bax2U8bgv1ZVy0OpoHHZyz3X5xEW
ISFzom9PqRUqzyM0Brn-BPb2-1-CovE-27-a-0LCC... 
```

add key to github

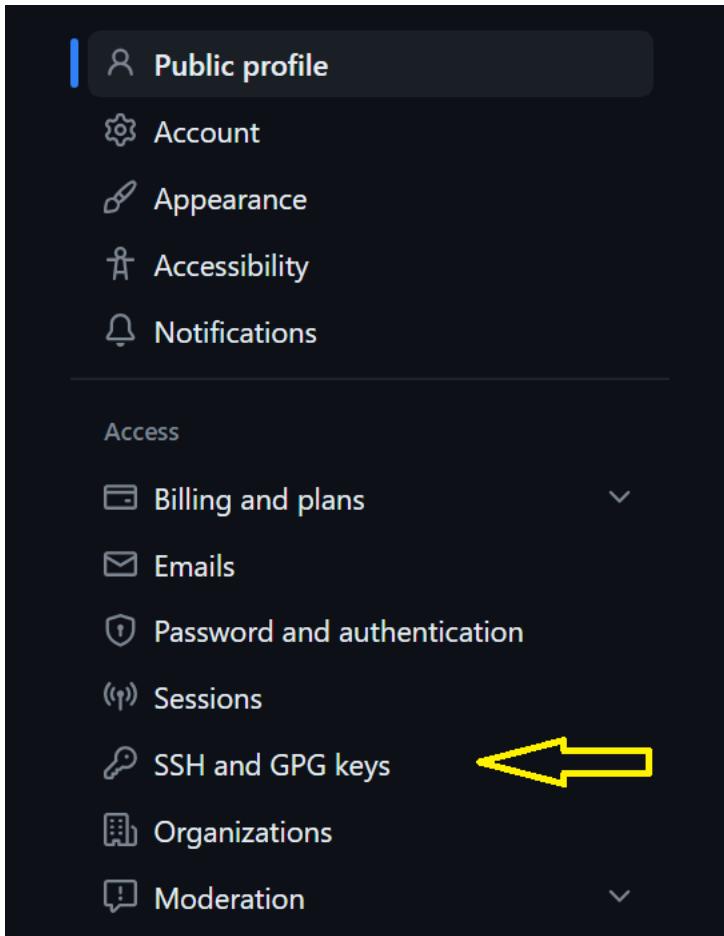
Open the GitHub website and log in to your account. Go to the top right hand icon.



Go to settings.



Click SSH and GPG keys



Add SSH key.

The page title is "SSH keys". A message states: "There are no SSH keys associated with your account." Below it, there is a link to "Check out our guide to generating SSH keys or troubleshoot common SSH problems." A green button labeled "New SSH key" is highlighted with a yellow box.

SSH keys / Add new

Title

Key type

Authentication Key

Key

```
ssh-rsa
AAAB3NzaC1yc2EAAAQABAAQAC7U5EF5Y/IVuSEnBxKoVU8fu8YU4/jX4xv1DDvDtv+aHukHNi9vHqwlwDP1Bax2U8
bgv1ZVyQOp0HHZyz3X5xEW4hUsRrISFzom9PqRUqzyM0Bm+hBP2p1wGpvEm27na0kCGyg6wwjxmaK4fVvq7MTquB+Hb
W12/cIkDOJw8sdztYMtrYcJFaK2W7loLB0odl6teQbpzf0dHXS/AHPaupxo7O4z2jM0aw+6w3K2Td7OUp4LfgukkwXWe6KXI
WGjkdjwLDawiDfrOsAoSHbqauOa6qxT58iOzpnfiHFgT+oScvLNsm1lecTrmN534aPsApo5wx3g3nElZRIRPKmMN+HXQuYlx
hXrChbz15kwM1mGT[REDACTED]3q/FqJTI8yemF3fJWCFCULIXUxTGV4/xV51hFta6voHW
OQsLXulgaVCUYC+VNai4MmGwSxRCLALhESTb+xCEHO1i1XQqVnDrrvRQPQ0+eh27Tbm0fViEA8mGFT6udC2oD+zQwhh
GPXYJwDA6+JLQSiNYLSP3sz6b3wHGlrvmKUsR+IloP9J0M5z9yURxcZvQlmxv9PHFU1F+6Q1je0xtc89KFfvLCJpnSp9na5r9A
VXw93H9JT+prSTaAFuuQSuk9kv45QEZZQaqPXyWrE+lyf3G9P0LTQ== miroslavkrajca@sunnyorange.edu
```

Add SSH key

SSH keys

New SSH

This is a list of SSH keys associated with your account. Remove any keys that you do not recognize.

Authentication Keys

 web1	SHA256:1/z71STtTPy5G6bMHPok9J0rzuHNM1NuTOXD7zjp+uU	Delete
SSH	Added on Jun 29, 2023	
	Never used — Read/write	

Check out our guide to [generating SSH keys](#) or troubleshoot [common SSH problems](#).

verify key

```
(orange) occc@occc-VirtualBox:/csc108/project1/mywebsite$ ssh -T git@github.com
Hi computersciencedept! You've successfully authenticated, but GitHub does not provide shell access.
(orange) occc@occc-VirtualBox:/csc108/project1/mywebsite$
```

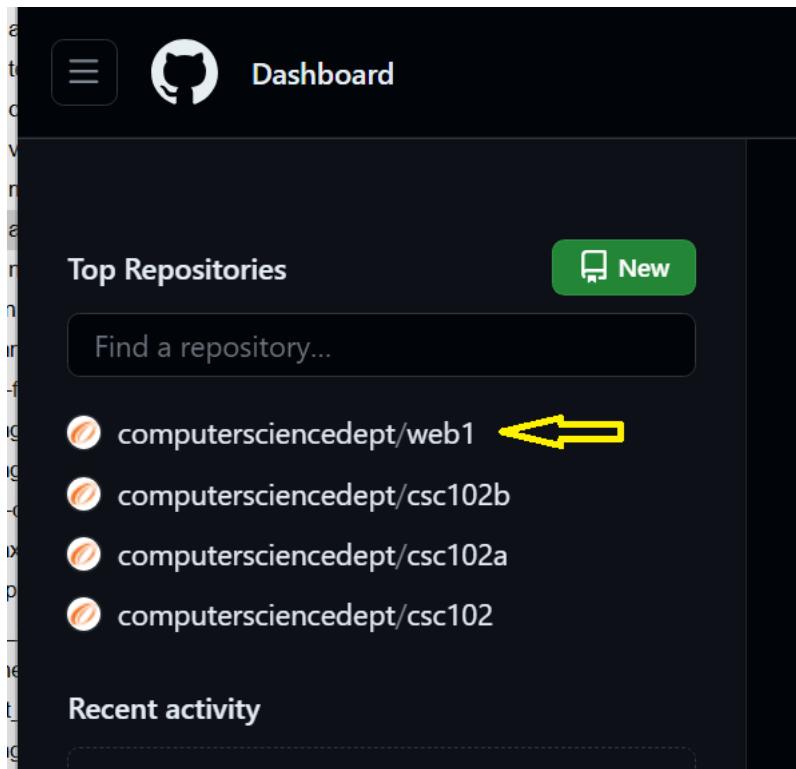
9.8.3 Finish repo upload

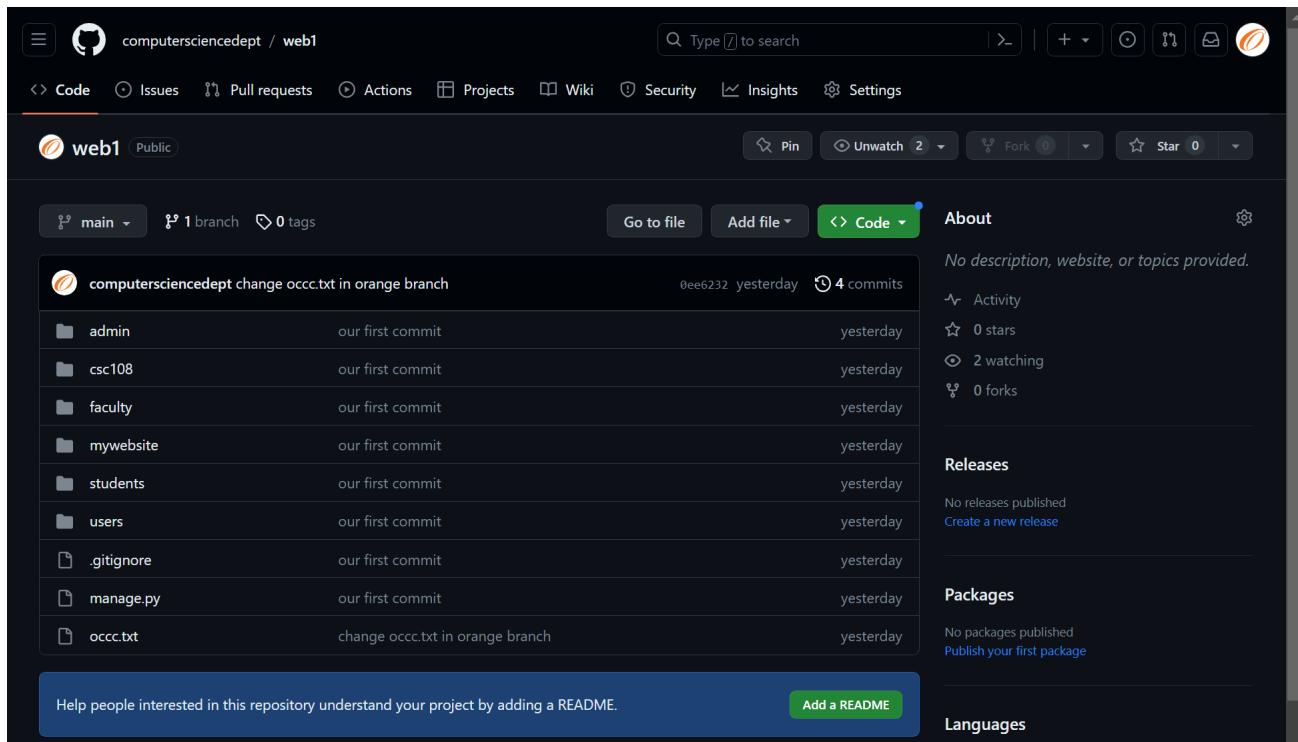
```
git remote add origin git@github.com:computersciencedept/web1.git
```

```
git push -u origin main
```

```
oxxx@oxxx-VirtualBox:/csc108/project1/mywebsite$ git remote add origin git@github.com:computersciencedept/web1.git
error: remote origin already exists.
(oxxx) oxxx@oxxx-VirtualBox:/csc108/project1/mywebsite$ git push -u origin main
Enumerating objects: 487, done.
Counting objects: 100% (487/487), done.
Compressing objects: 100% (465/465), done.
Writing objects: 100% (487/487), 3.71 MiB | 4.30 MiB/s, done.
Total 487 (delta 124), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (124/124), done.
To github.com:computersciencedept/web1.git
 * [new branch]      main -> main
Branch 'main' set up to track remote branch 'main' from 'origin'.
(oxxx) oxxx@oxxx-VirtualBox:/csc108/project1/mywebsite$
```

go to github and verify upload of repo





9.8.4 Django requirements file

A Python requirements file is a simple text file that lists all of the Python package dependencies a Python project needs to run correctly. Each line of the file includes one package. The file is typically called requirements.txt.

When creating Python applications, it's common to include a requirements.txt file in the root directory of the project. This allows other developers to understand what dependencies are required and to install them all at once using pip, the Python package installer. The command to do so is pip install -r requirements.txt.

Each line in the requirements file normally has the format <package-name>==<package-version>. This ensures not only that the correct packages are installed, but also the correct versions of these packages. However, the ==<package-version> part is optional, and if it's omitted, the latest version of the package will be installed.

Creating a requirements.txt file manually can be quite laborious, especially for large projects with many dependencies. Fortunately, pip can generate a requirements.txt file for you based on your current Python environment.

You can use the pip freeze command to list all the installed packages and their

versions in your current Python environment. You can then redirect this output to a file called requirements.txt.

Here's how you would do this in a command prompt or terminal:

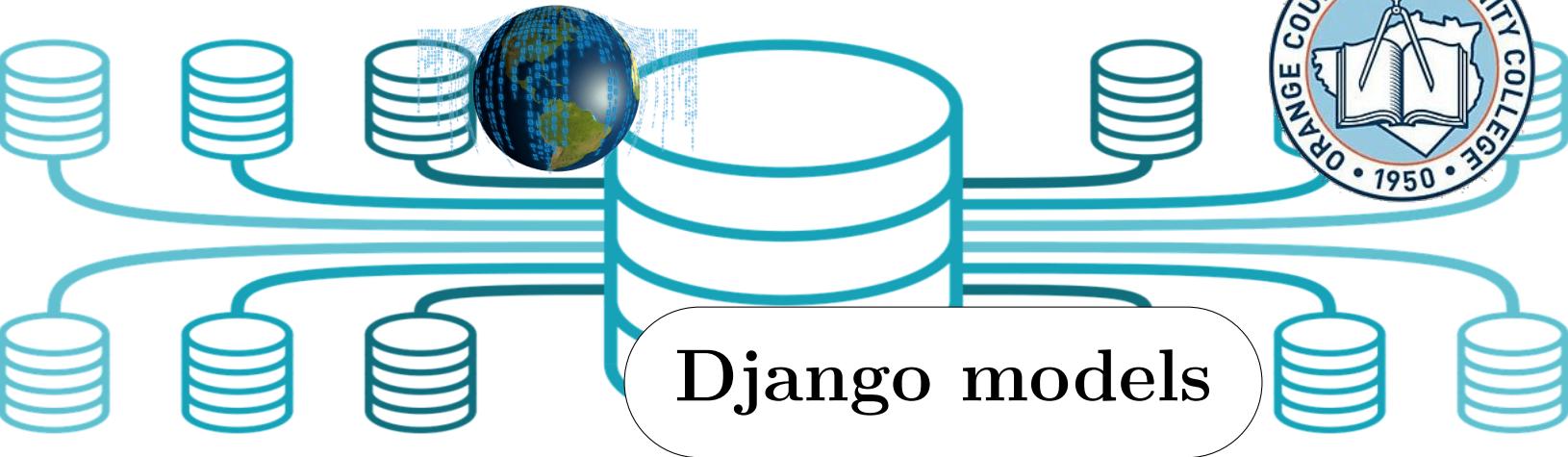
```
pip freeze > requirements.txt
```

This will create a requirements.txt file in the current directory. The file will contain a list of all installed packages in the current Python environment, along with the specific versions installed.

This approach, however, lists all installed packages, which might be more than you need for your specific project. If you're working on a specific project, it's often a good idea to use a virtual environment (such as those created with venv or conda). This way, you can keep your project's dependencies isolated from other Python projects and system Python packages. You can then use pip freeze in this isolated environment to get a list of just the dependencies your project needs.

Please note, the pip freeze method might not capture packages installed via methods other than pip (like apt-get, yum, etc.), so in such cases, manual editing of the requirements.txt might be needed.

```
(orange) occc@occc-VirtualBox:/csc108/project1/mywebsite$ pip freeze > requirements.txt
(orange) occc@occc-VirtualBox:/csc108/project1/mywebsite$ more requirements.txt
asgiref==3.7.2
certifi==2023.5.7
charset-normalizer==3.1.0
crispy-bootstrap5==0.7
defusedxml==0.7.1
Django==4.2.2
django-crispy-forms==2.0
django-debug-toolbar==4.1.0
django-extensions==3.2.3
gunicorn==20.1.0
httpie==3.2.2
idna==3.4
markdown-it-py==3.0.0
mdurl==0.1.2
multidict==6.0.4
psycopg2-binary==2.9.6
pydot==1.4.2
Pygments==2.15.1
pyparsing==3.1.0
PySocks==1.7.1
requests==2.31.0
requests-toolbelt==1.0.0
rich==13.4.2
sqlparse==0.4.4
typing_extensions==4.6.3
urllib3==2.0.3
(orange) occc@occc-VirtualBox:/csc108/project1/mywebsite$ █
```



10.1 ➤ what is model ?

In Django, a model is the single, definitive source of information about your data. It contains the essential fields and behaviors of the data you're storing. Generally, each model maps to a single database table.

Each model is a Python class that subclasses `django.db.models.Model`, and each attribute of the model represents a database field or a relation.

The basics:

- Each model is a Python class that subclasses `django.db.models.Model`.
- Each attribute of the model represents a database field or a relation.
- With all of this, Django gives you an automatically-generated database-access API

Models are defined in the `app/models.py`

Example of a simple model from django tutorial.

example model.py

```

1 from django.db import models
2
3
4 class Person(models.Model):
5     first_name = models.CharField(max_length=30)
6     last_name = models.CharField(max_length=30)

```

first_name and last_name are fields of the model. Each field is specified as a class attribute, and each attribute maps to a database column.

The above Person model would create a database table like this:

create table from model

```

1 CREATE TABLE myapp_person (
2     "id" bigint NOT NULL PRIMARY KEY GENERATED BY DEFAULT AS IDENTITY,
3     "first_name" varchar(30) NOT NULL,
4     "last_name" varchar(30) NOT NULL
5 );

```

Be careful not to choose field names that conflict with the models API like clean, save, or delete.

10.2 field types

<https://docs.djangoproject.com/en/4.2/ref/models/fields/#field-types>

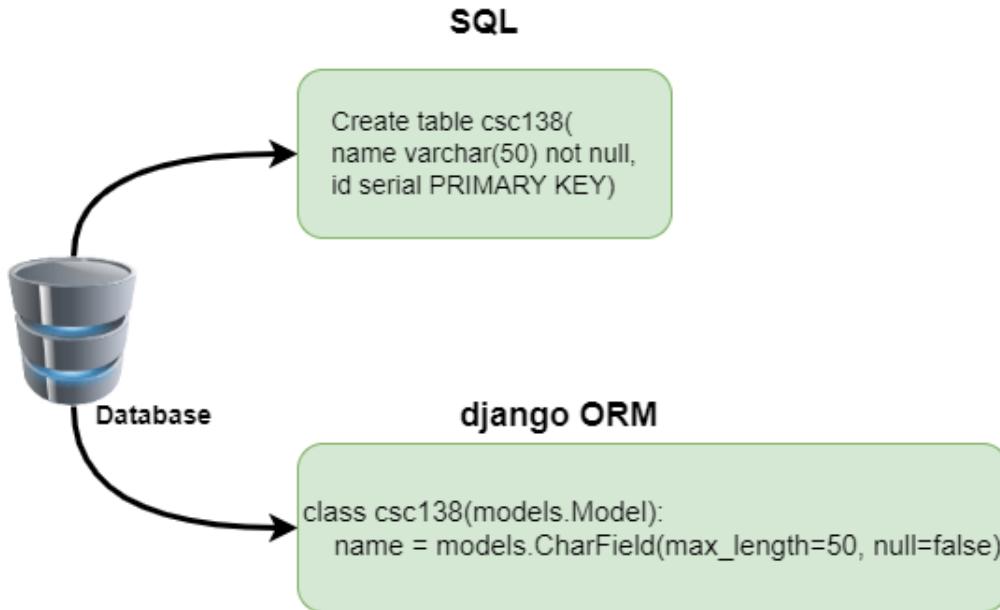
10.3 ORM

ORM stands for Object-Relational Mapping. It's a programming technique that allows you to interact with your database, like you would with SQL. In other words, it's a way to create, retrieve, update, and delete records in your database using Python code instead of writing SQL queries.

Django comes with its own ORM. It's a big part of what makes Django a powerful and flexible framework.

The Django ORM has several advantages:

- **Abstraction:** You can change your database anytime. If you are using MySQL and decide to switch to PostgreSQL, all you need to change is a single line in your settings. You don't have to rewrite your queries.
- **Ease of use:** You don't have to write SQL queries. Django ORM allows you to interact with your database like you would with Python Objects.
- **Security:** Django ORM provides a secure interface to the database, reducing common security mistakes developers can make.
- **Consistency:** SQL is inconsistent across different databases. You want to describe your data once, not create separate sets of SQL statements for each database to which the application will be deployed.
- **Avoids Introspection:** Any application has to know something about the underlying database structure. There are only two ways to do that: have an explicit description of the data in the application code, or introspect the database at runtime. As introspection has a high overhead and is not perfect, Django's creators chose the first option.
- **Version Control:** Storing models in your codebase makes it easier to keep track of design changes.
- **Advanced Metadata:** Having models described in code rather than SQL allows for special data-types (e.g., email addresses), and provides the ability to store much more metadata than SQL.



10.3.1 different types of ORM

There are several different types of Object-Relational Mapping (ORM) systems, which can be categorized according to the methods they use to implement object-relational mapping.

Here are three main types:

- **Active Record Pattern:** In this pattern, a model instance represents a database row. The database is wrapped by the business object, which takes care of encapsulating the database access. The object's member variables represent the fields of the row in the database and instance methods do the database operations. Active Record usually allows for simpler and more straightforward access to data, but it might not be as flexible for complex scenarios. Ruby on Rails' Active Record and Laravel's Eloquent ORM are examples of this pattern.
- **Data Mapper Pattern:** This pattern separates the in-memory objects from the database, using a layer of Mappers that move data between objects and database records. This allows the model and database schemas to evolve independently of each other. It's a bit more complex but also more flexible, especially for complex domain logic or situations where you don't have full control over your database schema. The Java framework Hibernate and the Python library SQLAlchemy are examples of this pattern.
- **Hybrid Approach:** Some ORMs, including Django's ORM, combine aspects of both the Active Record and Data Mapper patterns. These ORMs provide a simple API for basic database operations but also allow for more complex mappings when necessary.

The type of ORM that is best suited for a particular application depends on the specific requirements and constraints of that application, such as the complexity of the domain model, the expected size and performance of the database, and the experience and preferences of the development team.

10.4 ➤ migrations

<https://docs.djangoproject.com/en/4.2/topics/migrations/> Migrations are Django's way of propagating changes you make to your models (adding a field, deleting a model, etc.) into your database schema. They're designed to be mostly automatic, but you'll need to know when to make migrations, when to run them, and the common problems you might run into.

There are several commands which you will use to interact with migrations

and Django's handling of database schema:

- **migrate** , which is responsible for applying and unapplying migrations.
 - **makemigrations** , which is responsible for creating new migrations based on the changes you have made to your models.
 - **sqlmigrate** , which displays the SQL statements for a migration.
 - **showmigrations** , which lists a project’s migrations and their status.

Documentation on above commands:

migrate

makemigrations

sqlmigrate

showmigrations

You should think of migrations as a version control system for your database schema. `makemigrations` is responsible for packaging up your model changes into individual migration files - analogous to commits - and `migrate` is responsible for applying those to your database.

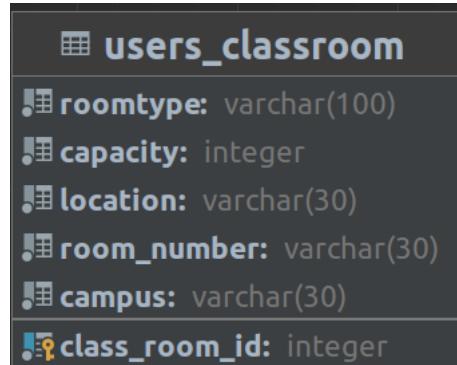
The migration files for each app reside in a “migrations” directory inside of that app, and are designed to be committed to, and distributed as part of, its codebase. You should be making them once on your development machine and then running the same migrations on your colleagues’ machines, your staging machines, and eventually your production machines.

classroom table model

```

13     location = models.CharField(max_length = 30)
14     room_number = models.CharField(max_length = 30)
15     campus = models.CharField(max_length = 30, choices=CAMPUS, default="↖
16         ↵ Middletown")
16     def __str__(self):
17         return str(self.room_number)

```



The table name will be a combination of <app name>_<model name>

roomtype = models.CharField

roomtype will be the table column and the models.CharField will be the data type. List of field types.

model-field-types

Each field will have options associated with that specific field.

field-options

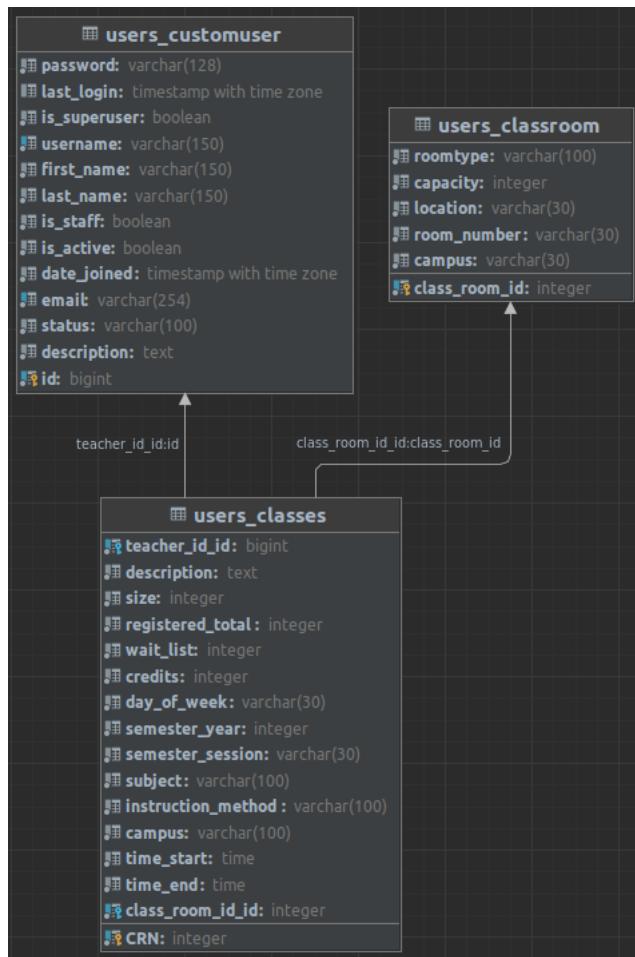
```

1  class classes(models.Model):
2      INSTRUCTION_METHOD = (
3          ("In-Person", "In-Person"),
4          ("Remote", "Remote"),
5          ("Off-Campus", "Off-Campus"),
6      )
7      CAMPUS = (
8          ("Middletown", "Middletown"),
9          ("Newburgh", "Newburgh"),
10         ("Other", "Other"),
11     )
12     SESSION = (
13         ("Fall", "Fall"),
14         ("Spring", "Spring"),
15         ("Summer-Full", "Summer-Full"),
16         ("Summer 1", "Summer 1"),

```

```
17     ("Summer 2", "Summer 2"),
18     ("Winter", "Winter"),
19     ("Other", "Other"),
20 )
21 class_room_id = models.ForeignKey(classroom,on_delete=models.DO_NOTHING,
22         )
22 CRN = models.AutoField(auto_created = True,
23                     primary_key = True,
24                     serialize = False,
25                     verbose_name ='class_room_id')
26 teacher_id = models.ForeignKey(CustomUser,on_delete=models.DO_NOTHING)
27 description = models.TextField()
28 title= models.CharField(max_length=20,default="")
29 size = models.IntegerField()
30 registered_total = models.IntegerField(default=0)
31 wait_list = models.IntegerField(default=0)
32 credits = models.IntegerField(default=0)
33 day_of_week = models.CharField(max_length = 30)
34 semester_year = models.IntegerField()
35 semester_session = models.CharField(max_length = 30,choices=SESSION)
36 subject = models.CharField(max_length = 100)
37 instruction_method = models.CharField(max_length = 100,choices=
38             INSTRUCTION_METHOD,default="In-Person")
39 campus = models.CharField(max_length = 100,choices=CAMPUS,default="✓
40             ↴ Middletown")
41 time_start = models.TimeField()
42 time_end = models.TimeField()
43
44 def __str__(self):
45     return str(self.CRN)
```

ERD diagram of the above model.



There are two fields that point (link) to other tables or in this case other models.

foreign keys

```

class_room_id = models.ForeignKey(classroom, on_delete=models.DO_NOTHING)
teacher_id = models.ForeignKey(CustomUser, on_delete=models.DO_NOTHING)
  
```

Since ORM is a layer on top of the SQL and database the Model is not pointing to a table but to another model class which is in turn pointing to the actual table.

```

class classes(models.Model):
    class_room_id = models.ForeignKey(classroom,on_delete=models.DO_NOTHING)
    CRN = models.AutoField(auto_created = True,
                          primary_key = True,
                          serialize = False,
                          verbose_name = "class_room_id")
    teacher_id = models.ForeignKey(CustomUser,on_delete=models.DO_NOTHING)
    description = models.TextField()
    title= models.CharField(max_length=20,default="")
    size = models.IntegerField()
    registered_total = models.IntegerField(default=0)
    wait_list = models.IntegerField(default=0)
    credits = models.IntegerField(default=0)
    day_of_week = models.CharField(max_length = 30)
    semester_year = models.IntegerField()
    semester_session = models.CharField(max_length = 30,choices=SESSION)
    subject = models.CharField(max_length = 100)
    instruction_method = models.CharField(max_length = 100,choices=INSTRUCTION_METHOD,default="In-Person")
    campus = models.CharField(max_length = 100,choices=CAMPUS,$,default="Middletown")
    time_start = models.TimeField()
    time_end = models.TimeField()

    def __str__(self):
        return str(self.CRN)

```

users_classes	
teacher_id_id:	bigint
description:	text
size:	integer
registered_total:	integer
wait_list:	integer
credits:	integer
day_of_week:	varchar(30)
semester_year:	integer
semester_session:	varchar(30)
subject:	varchar(100)
instruction_method:	varchar(100)
campus:	varchar(100)
time_start:	time
time_end:	time
class_room_id_id:	integer
CRN:	integer

```

class classroom(models.Model):
    CAMPUS = (
        ("Middletown","Middletown"),
        ("Newburgh","Newburgh"),
        ("Other","Other"),
    )
    roomtype = models.CharField(max_length = 100)
    capacity = models.IntegerField(validators =[validate_room_size])
    class_room_id = models.AutoField(auto_created = True,
                                    primary_key = True,
                                    serialize = False,
                                    verbose_name = "class_room_id")
    location = models.CharField(max_length = 30)
    room_number = models.CharField(max_length = 30)
    campus = models.CharField(max_length = 30,choices=CAMPUS,$,default="Middletown")
    def __str__(self):
        return str(self.room_number)

```

users_classroom	
roomtype:	varchar(100)
capacity:	integer
location:	varchar(30)
room_number:	varchar(30)
campus:	varchar(30)
class_room_id:	integer

ORM (Object-Relational Mapping) is a programming technique that allows developers to work with a database using an object-oriented paradigm rather than writing SQL queries directly. Here's a quick overview of how ORM relates to SQL:

- ORM libraries provide an abstraction layer between the application code and the database. Developers can then interact with database tables and records as objects/classes instead of using SQL.
- Behind the scenes, the ORM translates object-oriented operations into SQL queries. For example, saving an object in code might translate to an INSERT or UPDATE query. Loading objects from the database is usually done with a SELECT query.
- A key benefit of ORM is it reduces the amount of SQL a developer needs to write. Common CRUD operations can often be performed without any SQL using the ORM API.
- However, ORM does not eliminate the need for SQL entirely. Complex queries that cannot be expressed via the ORM may still require raw SQL. Database schema changes also often need DDL SQL statements.
- ORM libraries like Hibernate, SQLAlchemy, Django ORM etc provide features beyond basic query translation too - eager/lazy loading, identity mapping, caching, relationship handling etc.

So in summary, ORM provides an object interface to relational data, reduces boilerplate SQL statements, but still relies on SQL underneath for complex operations and schema changes. It essentially abstracts away the relational aspects for developers.

book author model

```
1 from django.db import models
2
3 class Author(models.Model):
4     name = models.CharField(max_length=100)
5
6 class Book(models.Model):
7     title = models.CharField(max_length=100)
8     author = models.ForeignKey(Author, on_delete=models.CASCADE)
9
10
11 # Create your models here.
```

book author example

```

1 >>> from dbexamples.models import Author, Book
2 >>> author = Author.objects.create(name="Miroslav Krajca")
3 >>> book = Book.objects.create(title="My Book", author_id=author.id)
4 >>> print(book.title, book.author_id)
5
6     My Book 1
7 >>> book2 = Book.objects.create(title="My Book occc", author_id=1) #set by ↴
8     ↴ id
9 >>> author.book_set.all()
10 <QuerySet [<Book: Book object (1)>, <Book: Book object (2)>]>
11 >>> print(author.name)
12     Miroslav Krajca
13 >>> for book_title in author.book_set.all():
14     ...     print(book_title.title)
15 ...
16 My Book
17 My Book occc
18 >>> one_book = Book.objects.get(title='My Book')
19 >>> print(one_book.author.name)
20     Miroslav Krajca
21 >>>

```

The `_set` attribute in Django models allows you to access related objects from the other side of a relationship.

Specifically:

- For a ForeignKey relationship from ModelA to ModelB, each instance of ModelA will have a `modelb_set` attribute to access the related ModelB instances.
- For a ManyToManyField, each instance of ModelA will have `modelb_set` to access the related ModelB instances, and vice versa.

query _set

```

1 # Author -> Book (ForeignKey)
2
3 author = Author.objects.get(name='John Doe')
4 author.book_set.all() # QuerySet of books by this author
5
6 # Book -> Author (reverse ForeignKey)
7
8 book = Book.objects.get(title='My Book')
9 book.author_set.all() # QuerySet with book's author
10
11 # Book <-> Author (ManyToManyField)
12
13 author = Author.objects.get(name='John Doe')
14 author.book_set.all() # Books by this author

```

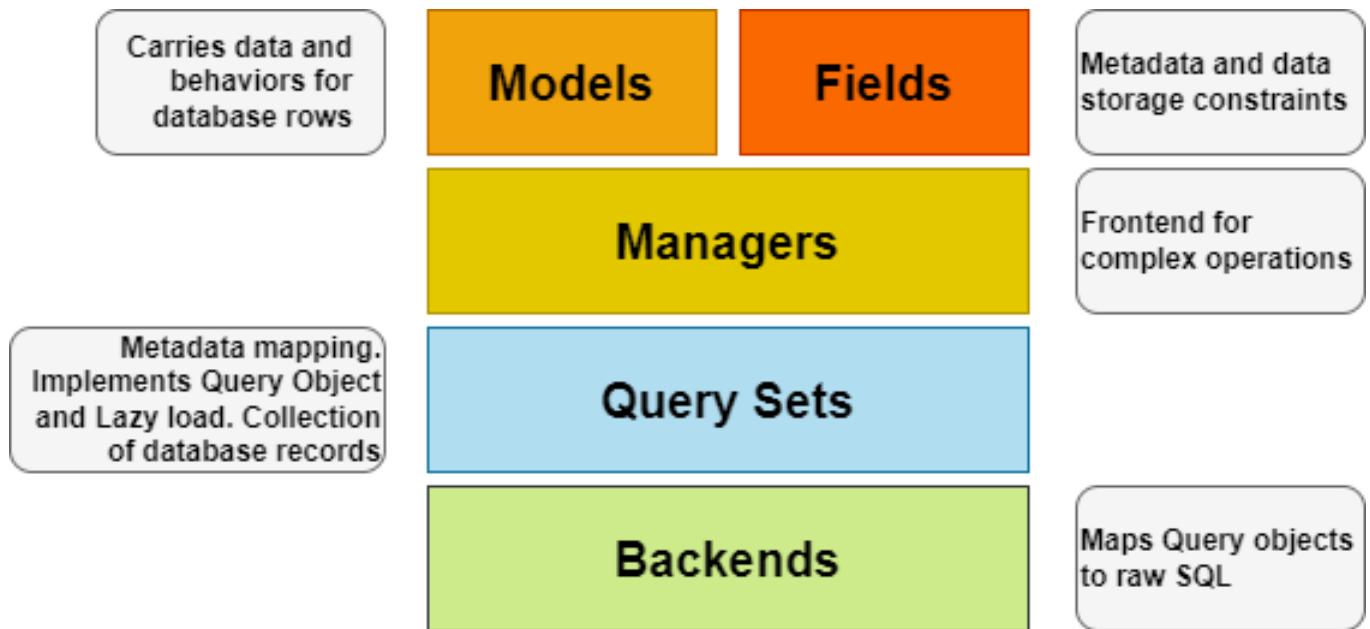
```
15  
16 book = Book.objects.get(title='My Book')  
17 book.author_set.all() # Authors of this book
```

So in summary:

`_set` allows traversing from one side of a relationship to the other. It generates the default related name that Django uses. Very useful for querying related objects based on the foreign key or many-to-many fields.



Querysets



We have already seen models. They map field names to datatypes as well as to other models for relations. They also have other constraints such as alternate names, default values, table names, auto creation of primary keys(if we want to),...

11.0.1 Managers

A manager in Django is an interface through which database query operations are provided to Django models.

Some key points about managers:

- Each Django model has at least one manager specified. The default manager is called **objects**.
- Managers query the database to return instances of the model. For example:

all()

```
1 MyModel.objects.all()
```

This queries the database and returns all MyModel instances.

- Custom managers can be defined to add extra query methods. For example:

custom manager

```
1 class BookManager(models.Manager):
2     def get_recent_books(self):
3         return self.filter(pub_date__gte=timezone.now() - datetime.timedelta(days=30))
4
5 class Book(models.Model):
6     ...
7     objects = BookManager()
```

Now Book.objects.get_recent_books() can be used to get recent books.

- Managers also provide the interface for model creation. For example:

create

```
1 book = Book.objects.create(name='My Book', author='John')
```

- The default manager called **objects** is automatically created, but custom managers need to be explicitly specified.
- the default manager can be renamed just in case we have to use the objects as field name or we want to call it something else.

rename default manager

```
1 from django.db import models
2
3 class Person(models.Model):
4     ...
5     people = models.Manager()
```

managers

In summary, managers provide the interface to query and create model instances from the database. They are an important part of how models communicate with the underlying data.

11.0.2 backends

The database backend abstraction allows Django to be independent of the actual database used. The same Django ORM code will work with different backends like SQLite, MySQL and PostgreSQL.

11.1 QuerySet API

Usually when you'll interact with a QuerySet you'll use it by chaining filters. To make this work, most QuerySet methods return new querysets. These methods are covered in detail later in this section.

11.1.1 Methods that return new QuerySets

11.1.1.0 filter()

Returns a new QuerySet containing objects that match the given lookup parameters.

filter(*args, **kwargs)

The lookup parameters (**kwargs) should be in the format described in [Field lookups](#).
 Multiple parameters are joined via AND in the underlying SQL statement.
 If you need to execute more complex queries (for example, queries with OR statements), you can use

[Q objects](#).**filter**

```
cl = classes.objects.filter(title='CSC108')
```

classes.objects.filter(title='CSC108')

CRN	class_room_id	teacher_id	description	title	size	registered_total	wait_list	credits	day_of_week	semester_year
35	BT-357	mkrajca	Web Development	CSC108	20	0	0	4	T,R	2023

classes.objects.filter(title__startswith="CSC")

CRN	class_room_id	teacher_id	description	title	size	registered_total	wait_list	credits	day_of_week	semester_year
36	BT-355	mkrajca	Comp sci 1	CSC101	20	0	0	4	M,W	2023
35	BT-357	mkrajca	Web Development	CSC108	20	0	0	4	T,R	2023
38	BT-355	mkrajca	Scripting	CSC138	20	0	0	3	W	2023
37	BT-121	mkrajca	Scripting	CSC138	20	0	0	3	M,W	2023

11.1.1.0 exclude()

classes.objects.exclude(title__startswith="CI")

CRN	class_room_id	teacher_id	description	title	size	registered_total	wait_list	credits	day_of_week	semester_year
36	BT-355	mkrajca	Comp sci 1	CSC101	20	0	0	4	M,W	2023
35	BT-357	mkrajca	Web Development 1	CSC108	20	0	0	4	T,R	2023
46	BT-113	gthomas	Computer Forensics	CFR221	20	0	0	3	T,R	2023
56	TWR-210	cyang	Computer applications and graphics	EET110	20	0	0	3	W	2023
55	BT-357	cwarrington	Info Sec	CSS223	20	0	0	3	M,W,F	2023
38	BT-355	mkrajca	Scripting	CSC138	20	0	0	3	W	2023
37	BT-121	mkrajca	Scripting	CSC138	20	0	0	3	M,W	2023
41	NA	TBA	Place holder	TBA	20	0	0	0	M,T,W,R,F	9999

11.1.1.0 `annotate()`

Annotates each object in the QuerySet with the provided list of query expressions. An expression may be a simple value, a reference to a field on the model (or any related models), or an aggregate expression (averages, sums, etc.) that has been computed over the objects that are related to the objects in the QuerySet.

11.1.1.0 `alias()`

Same as `annotate()`, but instead of annotating objects in the QuerySet, saves the expression for later reuse with other QuerySet methods. This is useful when the result of the expression itself is not needed but it is used for filtering, ordering, or as a part of a complex expression. Not selecting the unused value removes redundant work from the database which should result in better performance.

11.1.1.0 `order_by()`

```
classes.objects.exclude(title__startswith="CI").order_by("title")
```

CRN	class_room_id	teacher_id	description	title	size	registered_total	wait_list	credits	day_of_week	semester_year	semester_session
46	BT-113	gthomas	Computer Forensics	CFR221	20	0	0	3	T,R	2023	
36	BT-355	mkrajca	Comp sci 1	CSC101	20	0	0	4	M,W	2023	
35	BT-357	mkrajca	Web Development 1	CSC108	20	0	0	4	T,R	2023	
38	BT-355	mkrajca	Scripting	CSC138	20	0	0	3	W	2023	
37	BT-121	mkrajca	Scripting	CSC138	20	0	0	3	M,W	2023	
55	BT-357	cwarrington	Info Sec	CSS223	20	0	0	3	M,W,F	2023	
56	TWR-210	cyang	Computer applications and graphics	EET110	20	0	0	3	W	2023	
41	NA	TBA	Place holder	TBA	20	0	0	0	M,T,W,R,F	9999	

11.1.1.0 reverse()

Use the `reverse()` method to reverse the order in which a queryset's elements are returned. Calling `reverse()` a second time restores the ordering back to the normal direction.

To retrieve the “last” five items in a queryset, you could do this:

```
my_queryset.reverse()[:5]
```

```
classes.objects.exclude(title__startswith="CI").order_by("title").reverse()
```

CRN	class_room_id	teacher_id	description	title	size	registered_total	wait_list	credits	day_of_week	semester_year
41	NA	TBA	Place holder	TBA	20	0	0	0	M,T,W,R,F	9999
56	TWR-210	cyang	Computer applications and graphics	EET110	20	0	0	3	W	2023
55	BT-357	cwarrington	Info Sec	CSS223	20	0	0	3	M,W,F	2023
38	BT-355	mkrajca	Scripting	CSC138	20	0	0	3	W	2023
37	BT-121	mkrajca	Scripting	CSC138	20	0	0	3	M,W	2023
35	BT-357	mkrajca	Web Development 1	CSC108	20	0	0	4	T,R	2023
36	BT-355	mkrajca	Comp sci 1	CSC101	20	0	0	4	M,W	2023
46	BT-113	gthomas	Computer Forensics	CFR221	20	0	0	3	T,R	2023

```
classes.objects.exclude(title__startswith="CI").order_by("title").reverse()[:4]
```

CRN	class_room_id	teacher_id	description	title	size	registered_total	wait_list	credits	day_of_week	semester_year
41	NA	TBA	Place holder	TBA	20	0	0	0	M,T,W,R,F	9999
56	TWR-210	cyang	Computer applications and graphics	EET110	20	0	0	3	W	2023
55	BT-357	cwarrington	Info Sec	CSS223	20	0	0	3	M,W,F	2023
38	BT-355	mkrajca	Scripting	CSC138	20	0	0	3	W	2023

11.1.1.0 **distinct()**

Returns a new QuerySet that uses SELECT DISTINCT in its SQL query. This eliminates duplicate rows from the query results.

```
classes.objects.filter(title__startswith="CS").distinct("title").reverse()
```

CRN	class_room_id	teacher_id	description	title	size	registered_total	wait_list	credits	day_of_week	semester_year
36	BT-355	mkrajca	Comp sci 1	CSC101	20	0	0	4	M,W	2023
35	BT-357	mkrajca	Web Development 1	CSC108	20	0	0	4	T,R	2023
38	BT-355	mkrajca	Scripting	CSC138	20	0	0	3	W	2023
55	BT-357	cwarrington	Info Sec	CSS223	20	0	0	3	M,W,F	2023

11.1.1.0 **values()**

Returns a QuerySet that returns dictionaries, rather than model instances, when used as an iterable.

Each of those dictionaries represents an object, with the keys corresponding to the attribute names of model objects.

```
classes.objects.filter(title__startswith="CS").distinct("title").reverse().values()
```

36	Comp sci 1	CSC101	20	0	0	4	M,W	2023
35	Web Development 1	CSC108	20	0	0	4	T,R	2023
38	Scripting	CSC138	20	0	0	3	W	2023
55	Info Sec	CSS223	20	0	0	3	M,W,F	2023

When using values() together with distinct(), be aware that ordering can affect the results.

Remember that dictionaries are not ordered in Python!!!!!!

11.1.1.0 **values_list()**

This is similar to values() except that instead of returning dictionaries, it returns tuples when iterated over. Each tuple contains the value from the respective field

or expression passed into the `values_list()` call — so the first item is the first field, etc.

11.1.1.0 `dates()`

Returns a QuerySet that evaluates to a list of `datetime.date` objects representing all available dates of a particular kind within the contents of the QuerySet.

11.1.1.0 `datetimes()`

Returns a QuerySet that evaluates to a list of `datetime.datetime` objects representing all available dates of a particular kind within the contents of the QuerySet.

11.1.1.0 `all()`

Returns a copy of the current QuerySet (or QuerySet subclass). This can be useful in situations where you might want to pass in either a model manager or a QuerySet and do further filtering on the result. After calling `all()` on either object, you'll definitely have a QuerySet to work with.

11.1.1.0 `union()`

Uses SQL's UNION operator to combine the results of two or more QuerySets. For example:

```
»> qs1 = Author.objects.values_list("name")
»> qs2 = Entry.objects.values_list("headline")
»> qs1.union(qs2).order_by("name")
```

11.1.1.0 `intersection()`

Uses SQL's INTERSECT operator to return the shared elements of two or more QuerySets.

11.1.1.0 difference()

Uses SQL's EXCEPT operator to keep only elements present in the QuerySet but not in some other QuerySets.

11.1.1.0 select_related()

Returns a QuerySet that will “follow” foreign-key relationships, selecting additional related-object data when it executes its query. This is a performance booster which results in a single more complex query but means later use of foreign-key relationships won’t require database queries.

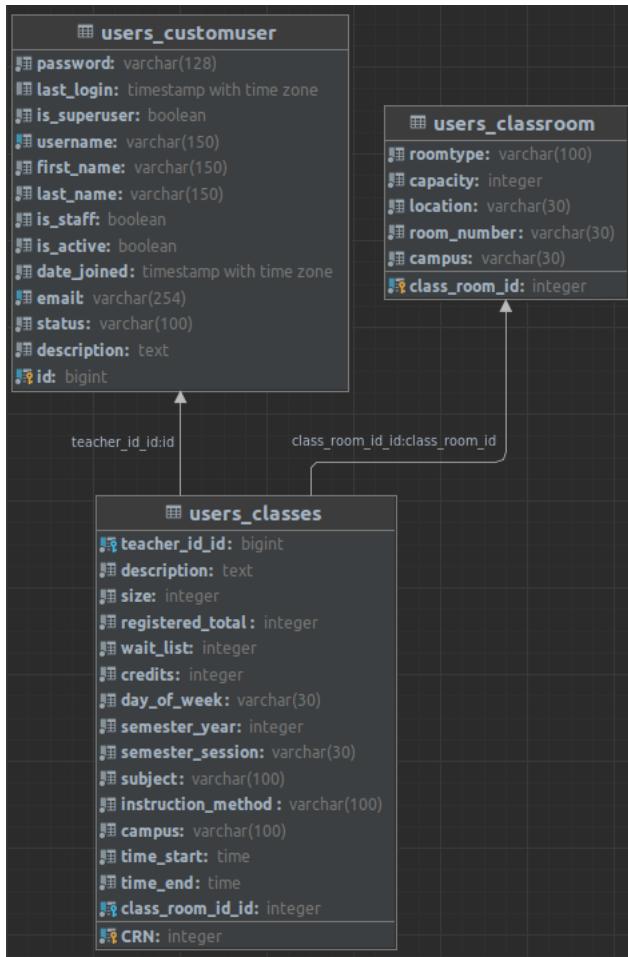
classess classroom models

```

1  class classroom(models.Model):
2      CAMPUS = (
3          ("Middletown", "Middletown"),
4          ("Newburgh", "Newburgh"),
5          ("Other", "Other"),
6      )
7      roomtype = models.CharField(max_length = 100)
8      capacity = models.IntegerField(validators =[validate_room_size])
9      class_room_id = models.AutoField(auto_created = True,
10          primary_key = True,
11          serialize = False,
12          verbose_name = 'class_room_id')
13      location = models.CharField(max_length = 30)
14      room_number = models.CharField(max_length = 30)
15      campus = models.CharField(max_length = 30, choices=CAMPUS,default=" ↴
16          ↴ Middletown")
17      def __str__(self):
18          return str(self.room_number)
19
20  class classes(models.Model):
21      INSTRUCTION_METHOD = (
22          ("In-Person", "In-Person"),
23          ("Remote", "Remote"),
24          ("Off-Campus", "Off-Campus"),
25      )
26      CAMPUS = (
27          ("Middletown", "Middletown"),
28          ("Newburgh", "Newburgh"),
29          ("Other", "Other"),
30      )
31
32  
```

```
29
30     )
31     SESSION = (
32         ("Fall", "Fall"),
33         ("Spring", "Spring"),
34         ("Summer-Full", "Summer-Full"),
35         ("Summer 1", "Summer 1"),
36         ("Summer 2", "Summer 2"),
37         ("Winter", "Winter"),
38         ("Other", "Other"),
39     )
40     class_room_id = models.ForeignKey(classroom, on_delete=models.DO_NOTHING)
41         ↴
42     CRN = models.AutoField(auto_created = True,
43         primary_key = True,
44         serialize = False,
45         verbose_name = 'class_room_id')
46     teacher_id = models.ForeignKey(CustomUser, on_delete=models.DO_NOTHING)
47     description = models.TextField()
48     title= models.CharField(max_length=20, default="")
49     size = models.IntegerField()
50     registered_total = models.IntegerField(default=0)
51     wait_list = models.IntegerField(default=0)
52     credits = models.IntegerField(default=0)
53     day_of_week = models.CharField(max_length = 30)
54     semester_year = models.IntegerField()
55     semester_session = models.CharField(max_length = 30, choices=SESSION)
56     subject = models.CharField(max_length = 100)
57     instruction_method = models.CharField(max_length = 100, choices=
58         ↴ INSTRUCTION_METHOD, default="In-Person")
59     campus = models.CharField(max_length = 100, choices=CAMPUS, default="",
60         ↴ Middletown")
61     time_start = models.TimeField()
62     time_end = models.TimeField()

63     def __str__(self):
64         return str(self.CRN)
```



```
cl = classes.objects.all()
print(cl)
for c in cl:
    print(c.title)
```

SQL queries from 1 connection

default 12.52 ms (3 queries)

Query	Timeline
<pre><code>SELECT "users_classes"."CRN", "users_classes"."class_room_id_id", "users_classes"."teacher_id_id", "users_classes"."description", "users_classes"."title", "users_classes"."size", "users_classes"."registered_total", "users_classes"."wait_list", "users_classes"."credits", "users_classes"."day_of_week", "users_classes"."semester_year", "users_classes"."semester_session", "users_classes"."subject", "users_classes"."instruction_method", "users_classes"."campus", "users_classes"."time_start", "users_classes"."time_end" FROM "users_classes"</code></pre>	
Connection: default Transaction status: Idle <pre><code>/dj/venv/lib/python3.10/site-packages/django/contrib/staticfiles/handlers.py in __call__(80) return self.application(environ, start_response) /dj/venv/lib/python3.10/site-packages/appmap/django.py in __call__(226) return self.get_response(request) /dj/first_site/registrar/views.py in db_tests(569) for c in cl:</code></pre>	
<pre><code>[+] SELECT ... FROM "django_session" WHERE ("django_session"."expire_date" > '2023-07-24T19:27:29.480339+00:00'::timestamptz AND "django_session"."session_key" = '8d40t5iucam80w390e41csa51av9xis9') LIMIT 21</code></pre>	
<pre><code>[+] SELECT ... FROM "users_customuser" WHERE "users_customuser"."id" = 5 LIMIT 21</code></pre>	

System check identified no issues (0 silenced).

July 24, 2023 - 19:27:26

Django version 4.2.3, using settings 'first_site.settings'

Starting development server at <http://127.0.0.1:8000/>

Quit the server with CONTROL-C.

CSC101

CSC108

CFR221

EET110

CSS223

CIT218

CIT117

CIT116

CIT112

CIT111

CIT100

CIT100

CIT100

CIT225

CIT105

CSC138

CSC138

TBA

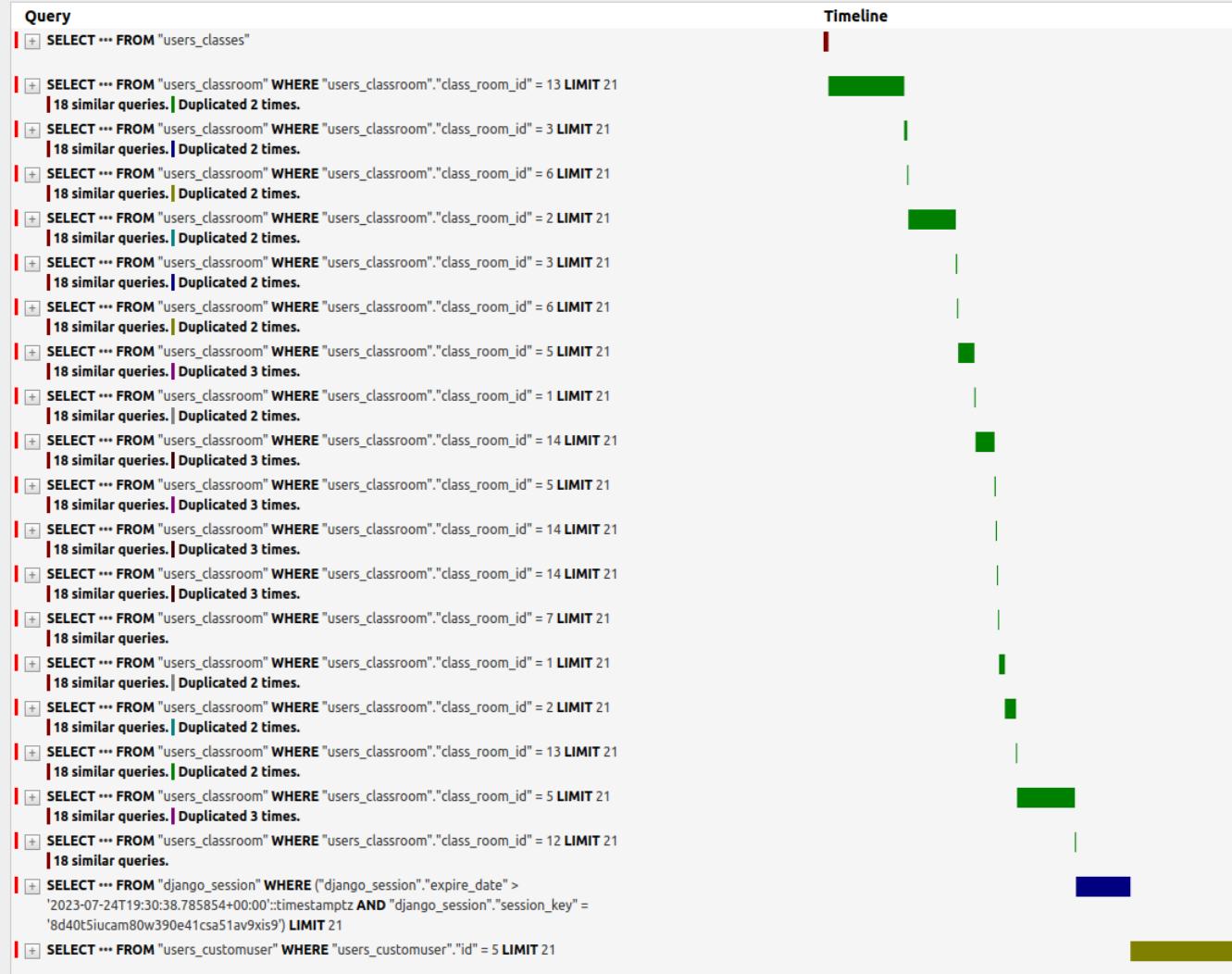
```

cl = classes.objects.all()
    print(cl)
    for c in cl:
        print(c.class_room_id)

```

SQL queries from 1 connection

default 45.08 ms (21 queries including 18 similar and 16 duplicates)



class_room_id is a foreign filed into another table. In this case it did a sql lookup for each classes room_id to resolve the data.

```
System check identified no issues (0 silenced).
July 24, 2023 - 19:30:31
Django version 4.2.3, using settings 'first_site.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CONTROL-C.
```

BT-355
BT-357
BT-113
TWR-210
BT-357
BT-113
BT-121
BT-255
BT-253
BT-121
BT-253
BT-253
TBA
BT-255
TWR-210
BT-355
BT-121
NA

```
cl = classes.objects.select_related('class_room_id').all()

for c in cl:
    print(c.class_room_id)
```

SQL queries from 1 connection

default 26.08 ms (3 queries)

Query	Timeline
<pre> SELECT "users_classes"."CRN", "users_classes"."class_room_id_id", "users_classes"."teacher_id_id", "users_classes"."description", "users_classes"."title", "users_classes"."size", "users_classes"."registered_total", "users_classes"."wait_list", "users_classes"."credits", "users_classes"."day_of_week", "users_classes"."semester_year", "users_classes"."semester_session", "users_classes"."subject", "users_classes"."instruction_method", "users_classes"."campus", "users_classes"."time_start", "users_classes"."time_end", "users_classroom"."class_room_id", "users_classroom"."roomtype", "users_classroom"."capacity", "users_classroom"."location", "users_classroom"."room_number", "users_classroom"."campus" FROM "users_classes" INNER JOIN "users_classroom" ON ("users_classes"."class_room_id_id" = "users_classroom"."class_room_id") </pre>	
Connection: default Transaction status: idle	
<pre> /dj/venv/lib/python3.10/site-packages/django/contrib/staticfiles/handlers.py in __call__(80) return self.application(environ, start_response) /dj/venv/lib/python3.10/site-packages/appmap/django.py in __call__(226) return self.get_response(request) /dj/first_site/registrar/views.py in db_tests(569) for c in cl: </pre>	
<pre> SELECT ... FROM "django_session" WHERE ("django_session"."expire_date" > '2023-07-24T19:35:20.775422+00:00':timestamptz AND "django_session"."session_key" = '8d40tSiucam80w390e41csa51av9xis9') LIMIT 21 </pre>	
<pre> SELECT ... FROM "users_customuser" WHERE "users_customuser"."id" = 5 LIMIT 21 </pre>	

```

SELECT "users_classes"."CRN",
"users_classes"."class_room_id_id",
"users_classes"."teacher_id_id",
"users_classes"."description",
"users_classes"."title",
"users_classes"."size",
"users_classes"."registered_total",
"users_classes"."wait_list",
"users_classes"."credits",
"users_classes"."day_of_week",
"users_classes"."semester_year",
"users_classes"."semester_session",
"users_classes"."subject",
"users_classes"."instruction_method",
"users_classes"."campus",
"users_classes"."time_start",

```

```
"users_classes"."time_end",
"users_classroom"."class_room_id",
"users_classroom"."roomtype",
"users_classroom"."capacity",
"users_classroom"."location",
"users_classroom"."room_number",
"users_classroom"."campus"
FROM "users_classes"
INNER JOIN "users_classroom"
ON ("users_classes"."class_room_id_id" = "users_classroom"."class_r
```

11.1.1.0 `prefetch_related()`

`prefetch_related(*lookups)` Returns a QuerySet that will automatically retrieve, in a single batch, related objects for each of the specified lookups.

This has a similar purpose to `select_related`, in that both are designed to stop the deluge of database queries that is caused by accessing related objects, but the strategy is quite different.

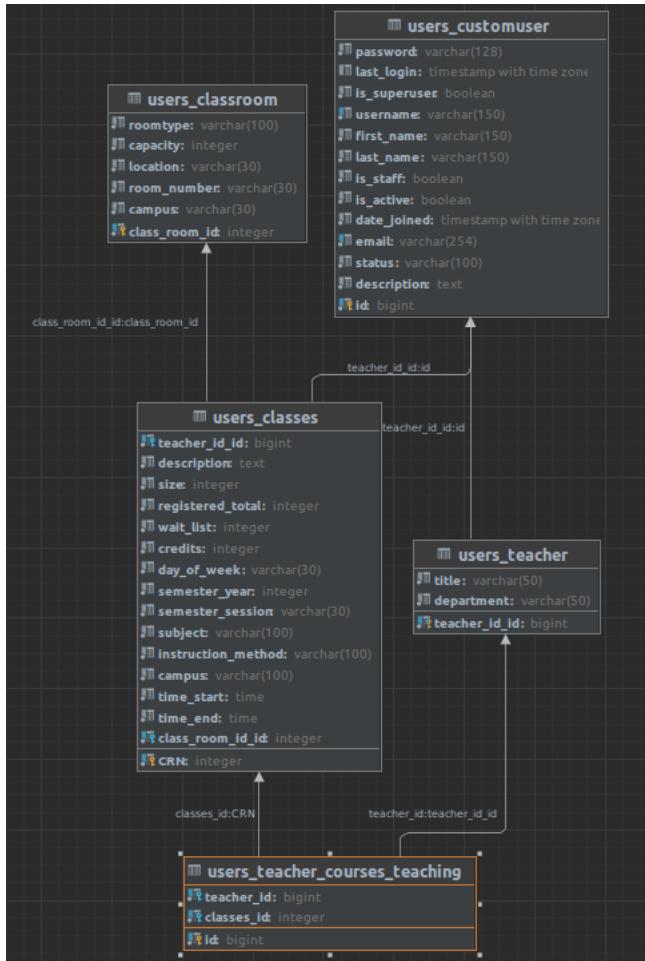
`select_related` works by creating an SQL join and including the fields of the related object in the SELECT statement. For this reason, `select_related` gets the related objects in the same database query. However, to avoid the much larger result set that would result from joining across a ‘many’ relationship, `select_related` is limited to single-valued relationships - foreign key and one-to-one.

`prefetch_related`, on the other hand, does a separate lookup for each relationship, and does the ‘joining’ in Python. This allows it to prefetch many-to-many and many-to-one objects, which cannot be done using `select_related`, in addition to the foreign key and one-to-one relationships that are supported by `select_related`. It also supports prefetching of `GenericRelation` and `GenericForeignKey`, however, it must be restricted to a homogeneous set of results. For example, prefetching objects referenced by a `GenericForeignKey` is only supported if the query is restricted to one `ContentType`.

```

1  class teacher(models.Model):
2      teacher_id = models.OneToOneField(CustomUser, on_delete=models.CASCADE,
3                                       primary_key=True, verbose_name='teacher_id')
4
5      title = models.CharField(max_length=50)
6      department = models.CharField(max_length=50, default='NA')
7      courses_teaching = models.ManyToManyField(classes)
8
9      def __str__(self):
10         return str(self.teacher_id)

```



From teachers to get courses teaching we have to use manytomany. A class can have several teachers assigned to them. A primary teacher and perhaps a teaching aide. A teacher can have multiple classes assigned to them. There is a many to many join table in between.

This will apply to a student as well. A student can be registered for many classes and a class can have multiple students.

```
teacher1 = teacher.objects.select_related('teacher_id').all()

for c in teacher1:
    temp = c.courses_teaching.all()
    print("-"*30)
    print(c.teacher_id.first_name)
    print("-"*30)
    #print(temp)
    for cla in temp:
        temp2 = cla.title
        print(temp2)
```

SQL queries from 1 connection

default 18.22 ms (10 queries including 7 similar)		
Query	Timeline	Time
<pre>+ SELECT ... FROM "users_teacher" INNER JOIN "users_customuser" ON ("users_teacher"."teacher_id_id" = "users_customuser"."id")</pre>		0.9
<pre>+ SELECT ... FROM "users_classes" INNER JOIN "users_teacher_courses_teaching" ON ("users_classes"."CRN" = "users_teacher_courses_teaching"."classes_id") WHERE "users_teacher_courses_teaching"."teacher_id" = 20</pre>		0.9
<pre> 7 similar queries.</pre>		
<pre>+ SELECT ... FROM "users_classes" INNER JOIN "users_teacher_courses_teaching" ON ("users_classes"."CRN" = "users_teacher_courses_teaching"."classes_id") WHERE "users_teacher_courses_teaching"."teacher_id" = 114</pre>		0.2
<pre> 7 similar queries.</pre>		
<pre>+ SELECT ... FROM "users_classes" INNER JOIN "users_teacher_courses_teaching" ON ("users_classes"."CRN" = "users_teacher_courses_teaching"."classes_id") WHERE "users_teacher_courses_teaching"."teacher_id" = 107</pre>		4.3
<pre> 7 similar queries.</pre>		
<pre>+ SELECT "users_classes"."CRN", "users_classes"."class_room_id_id", "users_classes"."teacher_id_id", "users_classes"."description", "users_classes"."title", "users_classes"."size", "users_classes"."registered_total", "users_classes"."wait_list", "users_classes"."credits", "users_classes"."day_of_week", "users_classes"."semester_year", "users_classes"."semester_session", "users_classes"."subject", "users_classes"."instruction_method", "users_classes"."campus", "users_classes"."time_start", "users_classes"."time_end" FROM "users_classes" INNER JOIN "users_teacher_courses_teaching" ON ("users_classes"."CRN" = "users_teacher_courses_teaching"."classes_id") WHERE "users_teacher_courses_teaching"."teacher_id" = 111 7 similar queries.</pre>		0.2
Connection: default		
Transaction status: Idle		
/dj/venv/lib/python3.10/site-packages/django/contrib/staticfiles/handlers.py in __call__(80) return self.application(environ, start_response)		
/dj/venv/lib/python3.10/site-packages/appmap/django.py in __call__(226) return self.get_response(request)		
/dj/first_site/registrar/views.py in db_tests(575) for cla in temp:		
<pre>+ SELECT ... FROM "users_classes" INNER JOIN "users_teacher_courses_teaching" ON ("users_classes"."CRN" = "users_teacher_courses_teaching"."classes_id") WHERE "users_teacher_courses_teaching"."teacher_id" = 113</pre>		0.2
<pre> 7 similar queries.</pre>		
<pre>+ SELECT ... FROM "users_classes" INNER JOIN "users_teacher_courses_teaching" ON ("users_classes"."CRN" = "users_teacher_courses_teaching"."classes_id") WHERE "users_teacher_courses_teaching"."teacher_id" = 115</pre>		0.2
<pre> 7 similar queries.</pre>		
<pre>+ SELECT ... FROM "users_classes" INNER JOIN "users_teacher_courses_teaching" ON ("users_classes"."CRN" = "users_teacher_courses_teaching"."classes_id") WHERE "users_teacher_courses_teaching"."teacher_id" = 112</pre>		0.2
<pre> 7 similar queries.</pre>		
<pre>+ SELECT ... FROM "django_session" WHERE ("django_session"."expire_date" > '2023-07-24T23:46:45.973208+00:00'::timestamptz AND "django_session"."session_key" = '8d40t5ucam80w390e41csa51av9xis9') LIMIT 21</pre>		5.6
<pre>+ SELECT ... FROM "users_customuser" WHERE "users_customuser"."id" = 5 LIMIT 21</pre>		5.2

System check identified no issues (0 silenced).

July 24, 2023 - 23:46:44

Django version 4.2.3, using settings 'first_site.settings'

Starting development server at http://127.0.0.1:8000/

Quit the server with CONTROL-C.

Miroslav

TBA
CSC108
CIT117
CIT116
CSC138
CSC138
CSC101

Bruce

TBA
CIT111
CIT100
CIT105

Cassius

TBA
EET110

TBA

CIT225

Russell

TBA
CIT225
CIT218

Giorgianni

TBA
CIT100
CFR221

Cartmell

TBA
CIT112
CIT100
CSS223

```
teacher1 = teacher.objects.prefetch_related('courses_teaching').all()

for c in teacher1:

    print("-"*30)
    #print(c.teacher_id)
    print("-"*30)
    #print(c.courses_teaching.all())
    for cla in c.courses_teaching.all():
        print(cla.title)
```

SQL queries from 1 connection

default 24.51 ms (4 queries)		Timeline	Time
Query			
<input type="checkbox"/> <code>SELECT "users_teacher"."teacher_id_id", "users_teacher"."title", "users_teacher"."department" FROM "users_teacher"</code>			0.4
Connection: default Transaction status: Idle			
/dj/venv/lib/python3.10/site-packages/django/contrib/staticfiles/handlers.py in __call__(80) return self.application(environ, start_response)			
/dj/venv/lib/python3.10/site-packages/appmap/django.py in __call__(226) return self.get_response(request)			
/dj/first_site/registrar/views.py in db_tests(569) for c in teacher1:			
<input type="checkbox"/> <code>SELECT ("users_teacher_courses_teaching"."teacher_id") AS "_prefetch_related_val_teacher_id", "users_classes"."CRN", "users_classes"."class_room_id_id", "users_classes"."teacher_id_id", "users_classes"."description", "users_classes"."title", "users_classes"."size", "users_classes"."registered_total", "users_classes"."wait_list", "users_classes"."credits", "users_classes"."day_of_week", "users_classes"."semester_year", "users_classes"."semester_session", "users_classes"."subject", "users_classes"."instruction_method", "users_classes"."campus", "users_classes"."time_start", "users_classes"."time_end" FROM "users_classes" INNER JOIN "users_teacher_courses_teaching" ON ("users_classes"."CRN" = "users_teacher_courses_teaching"."classes_id") WHERE "users_teacher_courses_teaching"."teacher_id" IN (115, 113, 114, 107, 112, 20, 111)</code>			6.7
Connection: default Transaction status: Idle			
/dj/venv/lib/python3.10/site-packages/django/contrib/staticfiles/handlers.py in __call__(80) return self.application(environ, start_response)			
/dj/venv/lib/python3.10/site-packages/appmap/django.py in __call__(226) return self.get_response(request)			
/dj/first_site/registrar/views.py in db_tests(569) for c in teacher1:			
<input checked="" type="checkbox"/> <code>SELECT ... FROM "django_session" WHERE ("django_session"."expire_date" > '2023-07-25T00:06:28.130327+00:00':timestamptz AND "django_session"."session_key" = '8d40t5iucam80w390e41csa51av9xis9') LIMIT 21</code>			12.0
<input checked="" type="checkbox"/> <code>SELECT ... FROM "users_customuser" WHERE "users_customuser"."id" = 5 LIMIT 21</code>			5.0

11.2 ➤ ORM for all teachers and classes

11.2.0.0 raw()

Takes a raw SQL query, executes it, and returns a django.db.models.query.RawQuerySet instance. This RawQuerySet instance can be iterated over just like a normal QuerySet to provide object instances.

raw sql.

template for raw query

```

1  {% extends "registrar/base.html" %} 
2  {% load static %} 
3  {% load crispy_forms_filters %} 
4  {% load crispy_forms_tags %} 
5  {% block title %} 
6      filter examples 
7  {% endblock title %} 
8  {% block custom_css %} 
9  {% endblock custom_css %} 
10 {% block content %} 
11     <span class="d-flex justify-content-center align-items-center input-group" 
12         -text"> 
13         <h1>direct cursor result</h1> 
14     </span> 
15     <table class="table table-striped"> 
16         <thead> 
17             <tr> 
18                 <th scope="col">title</th> 
19                 <th scope="col">first_name</th> 
20                 <th scope="col">last_name</th> 
21                 <th scope="col">Class_name</th> 
22                 <th scope="col">teacher_id</th> 
23             </tr> 
24         </thead> 
25         <tbody> 
26             {% for q in query2 %} 
27                 <tr> 
28                     <th scope="row">{{ q.title }}</th> 
29                     <td>{{ q.first_name }}</td> 
30                     <td>{{ q.last_name }}</td> 
31                     <td>{{ q.CLASS_NAME }}</td> 
32                     <td>{{ q.teacher_id_id }}</td> 
33                 </tr> 
34             {% endfor %} 
35         </tbody> 
36     <span class="d-flex justify-content-center align-items-center input-group" 
37         -text"> 
38         <h1>objects rawsql</h1> 
39     </span> 
40     <table class="table table-striped"> 
41         <thead> 
42             <tr> 
43                 <th scope="col">title</th> 
44                 <th scope="col">first_name</th> 
45                 <th scope="col">last_name</th> 
46                 <th scope="col">Class_name</th> 
47                 <th scope="col">teacher_id</th> 
48             </tr> 
49         </thead> 
50         <tbody> 
51             {% for q in query %} 
52                 <tr>

```

```

52         <th scope="row">{{ q.title }}</th>
53         <td>{{ q.first_name }}</td>
54         <td>{{ q.last_name }}</td>
55         <td>{{ q.CLASS_NAME }}</td>
56         <td>{{ q.teacher_id_id }}</td>
57     </tr>
58     {% endfor %}
59   </tbody>
60 </table>
61 {% endblock content %}
62 {% block custom_js %}
63 {% endblock custom_js %}

```

view for raw query

```

1 def db_tests(request):
2
3     field_list = []
4
5     with connection.cursor() as cursor:
6         cursor.execute("""SELECT
7             TEACHER.TITLE,
8             USERS.FIRST_NAME,
9             USERS.LAST_NAME,
10            CL.TITLE AS "CLASS_NAME",
11            TEACHER.teacher_id_id
12        FROM
13            USERS_CUSTOMUSER USERS
14        INNER JOIN USERS_TEACHER TEACHER ON TEACHER.TEACHER_ID_ID = USERS.ID
15        INNER JOIN USERS_TEACHER_COURSES_TEACHING TEACHING ON TEACHER. ↴
16            ↴ TEACHER_ID_ID = TEACHING.TEACHER_ID
17        INNER JOIN USERS_CLASSES CL ON TEACHING.CLASSES_ID = CL."CRN"
18        ORDER BY
19            USERS.ID;""")
20
21         columns = [col[0] for col in cursor.description]
22         dictionary_result = [dict(zip(columns, row)) for row in cursor. ↴
23             ↴ fetchall()]
24         cursor.close()
25         print(dictionary_result)
26         teacher1 = teacher.objects.raw("""select t.title, u.first_name, u. ↴
27             ↴ last_name, cl.title AS "CLASS_NAME", t.teacher_id_id
28        FROM users_customuser u
29        INNER JOIN users_teacher t ON t.teacher_id_id = u.id
30        INNER JOIN users_teacher_courses_teaching c ON t.teacher_id_id = c. ↴
31            ↴ teacher_id
32        INNER JOIN users_classes cl ON c.classes_id = cl."CRN"
33        ORDER BY t.teacher_id_id;""")
34
35         for c in teacher1:
36             print("{0}\t{1}\t{2}\t{3}\t{4}".format(c.title,c.first_name,c. ↴
37                 ↴ last_name,c.CLASS_NAME,c.teacher_id_id))

```

```

36
37     context = {"query":teacher1,
38                 "query2":dictionary_result,
39                 "field_list": field_list}
40     return render(request, 'registrar/filter.html', context)

```

direct cursor result

title	first_name	last_name	Class_name	teacher_id
Assistant Professor	Miroslav	Krajca	CIT116	20
Assistant Professor	Miroslav	Krajca	CIT117	20
Assistant Professor	Miroslav	Krajca	CSC101	20
Assistant Professor	Miroslav	Krajca	CSC138	20
Assistant Professor	Miroslav	Krajca	TBA	20
Assistant Professor	Miroslav	Krajca	CSC138	20
Assistant Professor	Miroslav	Krajca	CSC108	20
Adjunct	Cassius	Yang	EET110	107
Adjunct	Cassius	Yang	TBA	107
TBA	TBA	TBA	CIT225	111
Associate Professor	Cartmell	Warrington	CIT112	112
Associate Professor	Cartmell	Warrington	CSS223	112
Associate Professor	Cartmell	Warrington	CIT100	112
Associate Professor	Cartmell	Warrington	TBA	112
Instructor	Russell	Hammond	CIT225	113
Instructor	Russell	Hammond	TBA	113
Instructor	Russell	Hammond	CIT218	113
Department Chair	Bruce	Roman	CIT100	114
Department Chair	Bruce	Roman	CIT111	114
Department Chair	Bruce	Roman	CIT105	114
Department Chair	Bruce	Roman	TBA	114
Professor	Giorgianni	Thomas	CIT100	115
Professor	Giorgianni	Thomas	TBA	115
Professor	Giorgianni	Thomas	CFR221	115

objects rawsql

title	first_name	last_name	Class_name	teacher_id
Assistant Professor	Miroslav	Krajca	CIT116	20
Assistant Professor	Miroslav	Krajca	CIT117	20
Assistant Professor	Miroslav	Krajca	CSC101	20
Assistant Professor	Miroslav	Krajca	CSC138	20
Assistant Professor	Miroslav	Krajca	TBA	20
Assistant Professor	Miroslav	Krajca	CSC138	20
Assistant Professor	Miroslav	Krajca	CSC108	20
Adjunct	Cassius	Yang	EET110	107

11.3 ➤ passing data to a queryset on URL

urls with parameter

```

1 path("registrar/orm_teacher_class/",views.db_tests,name=" ↴
    ↴ orm_teacher_class_default"),
2 path("registrar/orm_teacher_class/<int:id>",views.orm_teacher_class,name=" ↴
    ↴ orm_teacher_class"),

```

view with parameter

```

1 def orm_teacher_class(request, id ):
2
3
4     teacher1 = teacher.objects.get(teacher_id=id)
5     context ={"query2":teacher1}
6     return render(request, 'registrar/orm_filter.html', context)

```

URL: http://csc108.sunyorange.edu.local:8000/registrar/orm_teacher_class/20

URL parameter passing

title	first_name	last_name	Class_name	teacher_id
Assistant Professor	Miroslav	Krajca	TBA	mkrajca
Assistant Professor	Miroslav	Krajca	CSC108	mkrajca
Assistant Professor	Miroslav	Krajca	CIT117	mkrajca
Assistant Professor	Miroslav	Krajca	CIT116	mkrajca
Assistant Professor	Miroslav	Krajca	CSC138	mkrajca
Assistant Professor	Miroslav	Krajca	CSC138	mkrajca
Assistant Professor	Miroslav	Krajca	CSC101	mkrajca

URL: http://csc108.sunyorange.edu.local:8000/registrar/orm_teacher_class/112

URL parameter passing

title	first_name	last_name	Class_name	teacher_id
Associate Professor	Cartmell	Warrington	TBA	cwarrington
Associate Professor	Cartmell	Warrington	CIT112	cwarrington
Associate Professor	Cartmell	Warrington	CIT100	cwarrington
Associate Professor	Cartmell	Warrington	CSS223	cwarrington



user data database

Before we model our database we should choose the right model.

- Flat Model
- Hierarchical Model
- Network Model
- Relational Model
- Star Schema
- Snowflake Schema

Flat model

A flat model database structure is a single, two-dimensional array where elements in each column are the same type of data, and elements in the same row relate to each other.

Think of this as a single, unrelated database table, like an Excel spreadsheet. If you run a small business with a handful of employees and want to store only their salary information, then a single, flat data model will suffice.

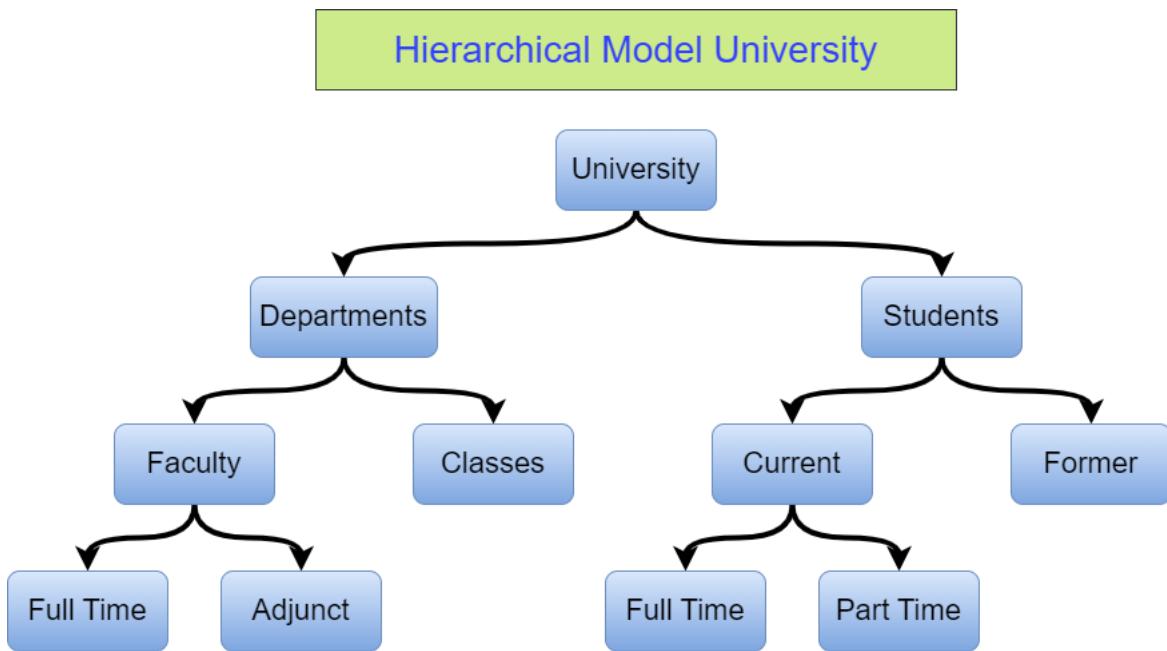
Flat Model Attendance

	Student1	Student2	Student3
Lecture1	X	✓	✓
Lecture2	✓	✓	✓
Lecture3	✓	✓	X

Hierarchical Model

Hierarchical database schemas have a tree-like structure, with a "root" node of data and child nodes that branch out from that root. There is a one-to-many relationship between parent and child nodes. This type of data schema is best reflected in XML or JSON files, where an entity can have sub-entities that are not shared with other entities.

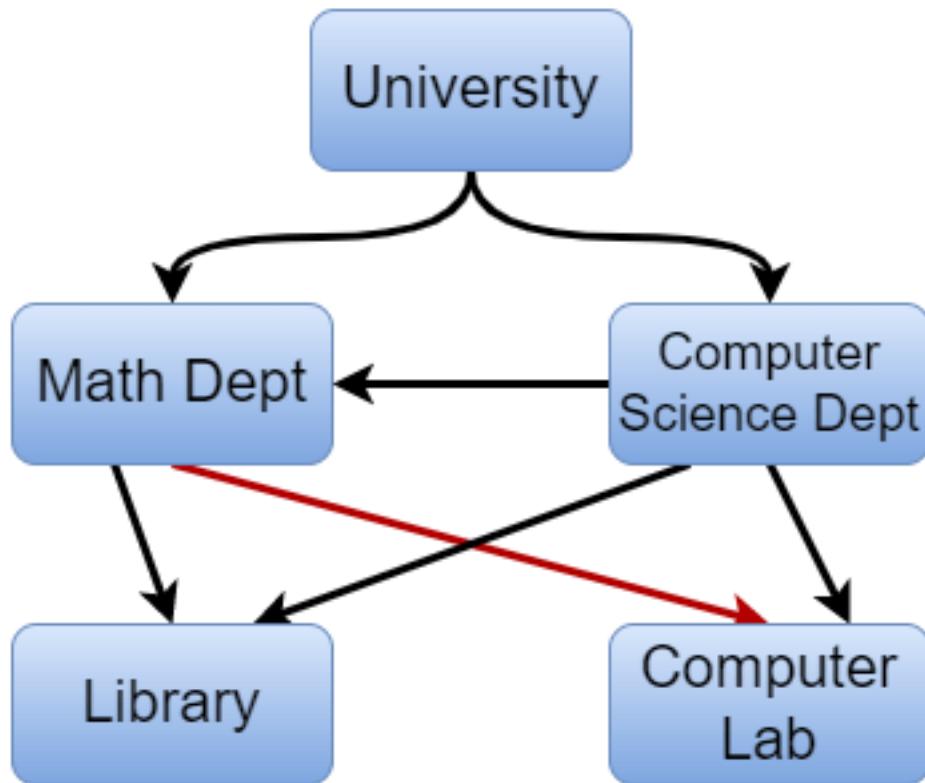
As the name suggests, the hierarchical database model is most appropriate for use cases in which the main focus of information gathering is based on a concrete hierarchy, such as several individual employees reporting to a single department at a company.



Network Model

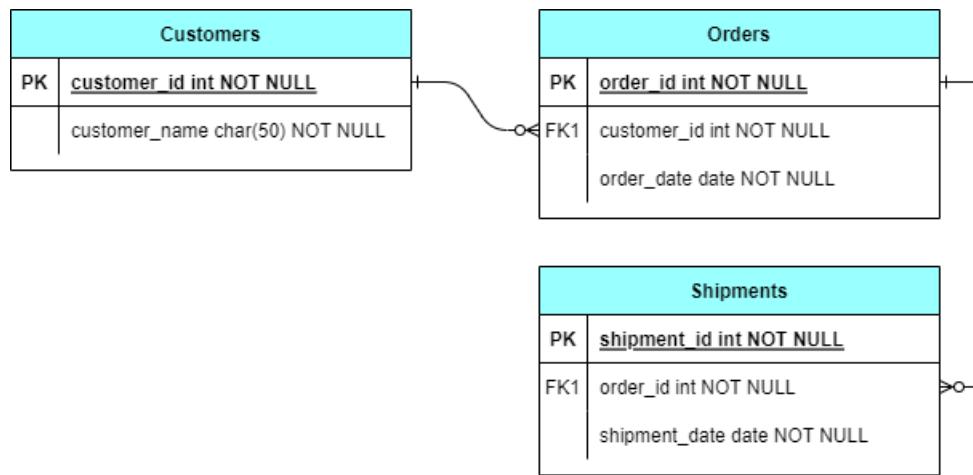
The network database model is a model for modeling the entities in such a way that one child entity can have more than one parent entity. In 1969, the network model was presented by Charles Bachman.

Network Model University



Relational Model

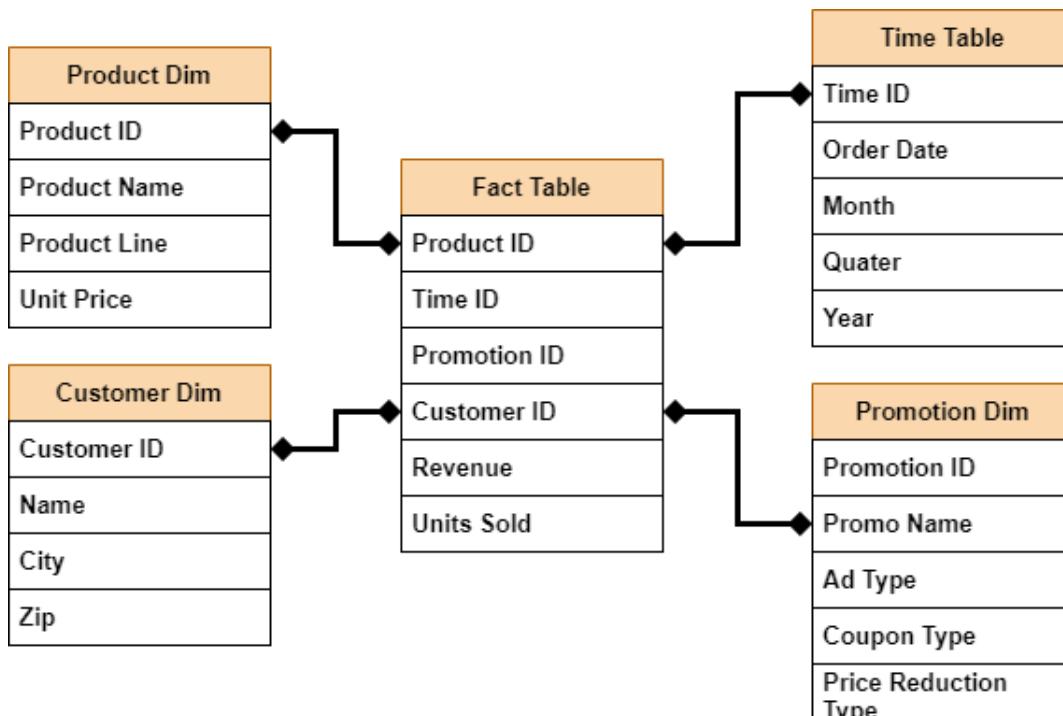
A relational database is a type of database that stores and provides access to data points that are related to one another. Relational databases are based on the relational model, an intuitive, straightforward way of representing data in tables. In a relational database, each row in the table is a record with a unique ID called the key. The columns of the table hold attributes of the data, and each record usually has a value for each attribute, making it easy to establish the relationships among data points.



Star Schema

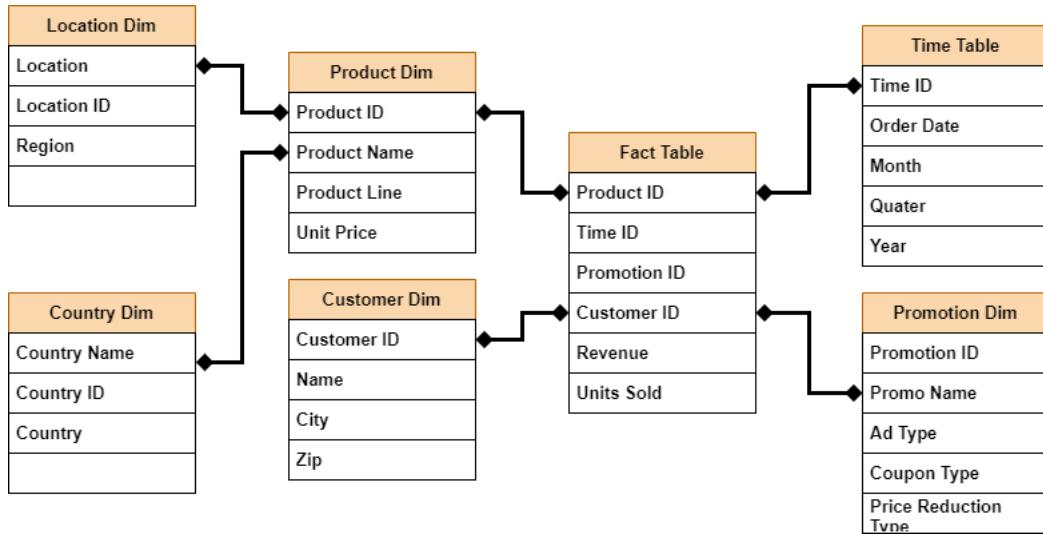
A star schema is a multi-dimensional data model used to organize data in a database so that it is easy to understand and analyze. Star schemas can be applied to data warehouses, databases, data marts, and other tools. The star schema design is optimized for querying large data sets.

Introduced by Ralph Kimball in the 1990s, star schemas are efficient at storing data, maintaining history, and updating data by reducing the duplication of repetitive business definitions, making it fast to aggregate and filter data in the data warehouse.



Snowflake Schema

As with the star schema, the snowflake schema has a central fact table that stores the main data points and references to its dimensional tables. Unlike the star schema, the snowflake schema dimensional tables can have their own dimensional tables, thus expanding how descriptive a dimension can be.



12.1 database model for college

12.1.1 model the database



The above model is still missing constraints such that we cannot put in a teacher_id

into classes for an instructor that might not exist. It is true that some classes can be created initially without a teacher assigned, however it should be a default placeholder teacher "tba". The above model can still have referential integrity problems down the road. We will refine the model as we make progress.

sql for db

```

1  create table classroom
2  (
3      "Classroom_id" integer not null
4          primary key,
5      room_type      text,
6      capacity       integer not null,
7      location       text
8  );
9
10 alter table classroom
11     owner to orangeuser;
12
13 create table classes
14 (
15     "CRN"           integer          not null
16         primary key,
17     "Classroom_id" integer default 0 not null
18         constraint classroom
19             references classroom,
20     teacher_id      integer default 0 not null,
21     description     text            not null,
22     size            integer default 20 not null,
23     registered_total integer default 0 not null,
24     time            text            not null,
25     credits          integer default 0 not null,
26     day_of_week     text            not null,
27     semester        text,
28     semester_year   integer,
29     semster_session integer,
30     "Subject"       text,
31     "Instruction_method" text,
32     campus          text,
33     time_start      time with time zone,
34     time_end        timestamp with time zone
35 );
36
37 alter table classes
38     owner to orangeuser;
39
40 create table users_customuser
41 (
42     id      bigint generated by default as identity
43         primary key,
44     password  varchar(128)          not null,
45     last_login timestamp with time zone,
46     is_superuser boolean          not null,
47     username   varchar(150)         not null
48         unique,
```

```
49      first_name  varchar(150)          not null,
50      last_name   varchar(150)          not null,
51      is_staff    boolean             not null,
52      is_active   boolean             not null,
53      date_joined timestamp with time zone not null,
54      email       varchar(254)
55          unique,
56      status      varchar(100)          not null,
57      description text               not null
58 );
59
60 alter table users_customuser
61     owner to orangeuser;
62
63 create table teacher
64 (
65     teacher_id integer not null
66         primary key
67         constraint teacher_id
68             references users_customuser,
69     title      text,
70     department text
71 );
72
73 alter table teacher
74     owner to orangeuser;
75
76 create table student
77 (
78     student_id  integer not null
79         primary key
80             references users_customuser,
81     courses_taken integer,
82     gpa        real,
83     credits    real,
84     major      text
85 );
86
87 alter table student
88     owner to orangeuser;
89
90 create index users_customuser_email_6445acef_like
91     on users_customuser (email varchar_pattern_ops);
92
93 create index users_customuser_username_80452fdf_like
94     on users_customuser (username varchar_pattern_ops);
95
96 create table student_classes
97 (
98     id        integer not null
99         primary key,
100    "CRN"     integer
101        constraint "CRN"
102            references classes,
103    student_id integer
104        constraint student_id
105            references student
```

```
106 );
107
108 alter table student_classes
109     owner to orangeuser;
110
111 create table courses_taken
112 (
113     student_id integer not null
114         primary key
115         constraint student_id
116             references student,
117     "CRN"    integer
118         constraint crn
119             references classes
120 );
121
122 alter table courses_taken
123     owner to orangeuser;
124
125 create table teacher_classes
126 (
127     teacher_id integer
128         constraint teacher_id
129             references teacher,
130     "CRN"    integer
131         constraint crn_id
132             references classes
133 );
134
135 alter table teacher_classes
136     owner to orangeuser;
```

12.1.2 Database design

- **Normalization:** It's a process that helps eliminate data redundancy and prevents issues like update anomalies. Normalization involves dividing large tables into smaller tables and defining relationships between them to increase clarity and simplify queries.
- **Consistent Data and Naming Conventions:** Consistency in data types, names, descriptions, and relationships across all tables can reduce confusion and improve the quality of the database.
- **Referential Integrity:** The relationships between tables must maintain consistency. This can be enforced using foreign keys, which help to

ensure that the relationship between two tables remains synchronized during updates and deletes.

- **Primary Keys and Indexes:** Every table should have a primary key that uniquely identifies each record. Indexes are also important for improving the speed of data retrieval operations on the database.
- **Scalability and Flexibility:** The database design should be scalable to handle increasing data volume. It should also be flexible enough to adapt to changing requirements.
- **Security:** It is important to have a database design that allows for appropriate security measures. This can include restricting access to certain data and encrypting sensitive information.
- **Performance:** Performance considerations should also be part of a good database design. This may involve optimizing query performance, considering the impact of transaction volume, and designing indexes for faster data retrieval.
- **Documentation:** A well-documented database design is much easier to maintain, understand, and modify. This should include diagrams showing table structures and relationships, as well as explanations of business rules and how the design meets those rules.
- **Usability:** The database should be easy to use so that users can query and update the database efficiently.
- **Business requirements:** Above all, a good database design should meet the needs of the business, both in terms of current requirements and anticipated future needs.

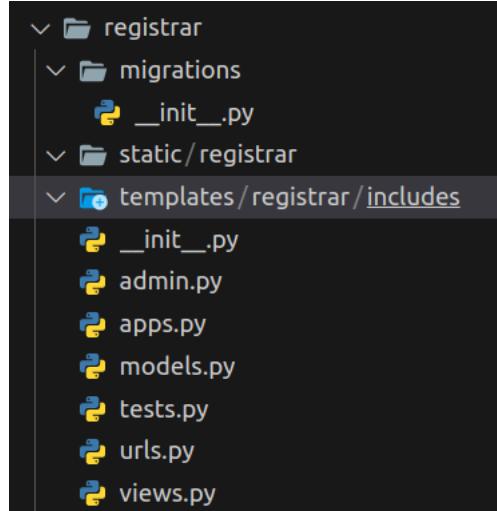
A good database design should always be based on thorough planning and analysis of the data needs of the organization or system. It may be beneficial to use Entity Relationship Diagrams (ERDs) or other modeling tools during the design process to help visualize the structure and relationships within the database.

12.2 ➤ College registrar page

The registrar page will be used by the registrar to create classes, assign teachers to classes, create students,... etc,etc.

After creating the registrar app our directory should look like this.

```
(venv) occc@occc-VirtualBox:/dj/first_site$ django-admin startapp registrar
```



We can recycle the templates from the csc108 app or just copy them to our new file system structure and edit for our purpose.

Make sure our app is registered.

register app in settings.py

```

1 # Application definition
2 INSTALLED_APPS = [
3     'django.contrib.admin',
4     'django.contrib.auth',
5     'django.contrib.contenttypes',
6     'django.contrib.sessions',
7     'django.contrib.messages',
8     'django.contrib.staticfiles',
9     'mainpage',

```

```
10     'users',
11     'faculty',
12     'registrar',
13 ]
```

registrar urls.py

```
1 from django.urls import path
2
3 from . import views
4
5 urlpatterns = [
6     path("registrar", views.home, name="registrar") ↴
7         ,
```

registrar views.py

```
1 from django.shortcuts import render
2 from django.http import HttpResponseRedirect
3 from django.template import loader
4 from django.contrib.auth.decorators import ↴
5     login_required
6
7
8 @login_required
9 def home(request):
10     return render(request, "registrar/home.html")
```

mainsettings urls.py

```
1 urlpatterns = [
2     path("csc108/", include("mainpage.urls")),
3     path('admin/', admin.site.urls),
4     path("", include("users.urls")),
5     path("", include('faculty.urls')),
```

```
6     path("", include('registrar.urls')),  
7 ]
```

12.2.1 Registrar base.html

registrar/templates/registrar/base.html

```
1  {% load static %}  
2  <!DOCTYPE html>  
3  <html lang="en-US">  
4      <head>  
5          <meta charset="utf-8">  
6          <meta name="description" content="CSC 108 Web programming 1">  
7          <meta name="keywords" content="Django, Suny Orange, CSC108">  
8          <meta name="author" content="Miroslav Krajca">  
9          <meta name="Classification" content='College'>  
10         <meta name="copyright" content='Miroslav Krajca'>  
11         <title>Registrar main web page</title>  
12         <link rel="stylesheet"  
13             type="text/css"  
14             href="{% static 'mainpage/bootstrap-5.3.0-dist/css/bootstrap.min.css' %}">  
15         <link rel="stylesheet"  
16             type="text/css"  
17             href="{% static 'registrar/navbar.css' %}">  
18     </head>  
19     <body>  
20         <main>  
21             {% include 'registrar/includes/navbar.html' %}  
22             <div style="width: 100%;">  
23                 {% include 'mainpage/includes/messaging.html' %}  
24                 {% block content %}{% endblock %}  
25             </div>  
26         </main>  
27         <!-- Optional Javascript -->  
28         <script src="{% static 'mainpage/jquery3.7.0/jquery-3.7.0.min.js' %}"></script>  
29         <script src="{% static 'mainpage/bootstrap-5.3.0-dist/js/bootstrap.bundle.min.js' %}"></script>  
30         <script src="{% static 'registrar/navbar.js' %}"></script>  
31         <script>  
32             $(document).ready(function() {  
33                 window.setTimeout(function() {  
34                     $(".alert").fadeTo(500, 0).slideUp(500, function() {  
35                         $(this).remove();  
36                     });  
37                 }, 5000);  
38             });  
39             $(function() {
```

```

41     $("li a").on("click", function() {
42         var url = $(this).attr("href");
43         $('[id*=dvDisplay]').html('<iframe class="row-container" ↴
44             ↴ frameborder="1" src="' + url + '" />');
45         return false;
46     });
47     </script>
48 </body>
49 </html>

```

registrar/templates/registrar/home.html

```

1  {% extends "registrar/base.html" %}
2  {% load static %}
3  {% block content %}
4      <div class="row-container">
5          <div class="first-row">
6              <p>
7                  <h1>
8                      You are logged in as <span style="color: red">{{ request.user }}<
9                          </span>
10                     </h1>
11                 </p>
12             <div id="dvDisplay" class="second-row"></div>
13         {% endblock content %}

```

registrar/templates/registrar/includes/navbar.html

```

1  {% load static %}
2  <svg xmlns="http://www.w3.org/2000/svg" style="display: none;">
3      <symbol id="bootstrap" viewBox="0 0 118 94">
4          <title>Bootstrap</title>
5          <path d="M0 2a2 2 0 0 1 2-2h12a2 2 0 0 1 2 2v12a2 2 0 0 1-2 2H2a2 2 0 0 ↴
6              ↴ 1-2-2V2zm15 2h-4v3h4V4zm0 4h-4v3h4V8zm0 4h-4v3h3a1 1 0 0 0 1-1v-2z
7              ↴ zm-5 3v-3H6v3h4zm-5 0v-3H1 v2a1 1 0 0 0 1 1h3zm-4-4h4V8H1v3zm0-4z
8              ↴ h4V4H1v3zm5-3v3h4V4H6zm4 4H6v3h4V8z" />
9          <path fill-rule="evenodd" clip-rule="evenodd" d="M24.509 0c-6.733 ↴
10             ↴ 0-11.715 5.893-11.492 12.284.214 6.14-.064 14.092-2.066 20.577C8.2
11             ↴ .943 39.365 5.547 43.485 0 44.014 v5.972c5.547.529 8.943 4.649 ↴
12             ↴ 10.951 11.153 2.002 6.485 2.28 14.437 2.066 20.577C12.794 88.106 ↴
13             ↴ 17.776 94 24.51 94H93.5c6.733 0 11.714-5.893 ↴
14             ↴ 11.491-12.284-.214-6.14.064-14.092 2.066-20.577 2.009-6.504 ↴
15             ↴ 5.396-10.624 10.943-11.153v-5.972c-5.547-5.29-8.934-4.649-10.943-11.153-2.002-6.484-2.28-14.437-2.066-20.577z
16             ↴ C105.214 5.894 100.233 0 93.5 0H24.508zM80 57.863C80 66.663 ↴
17             ↴ 73.436 72 62.543 72H44a2 2 0 0 1-2-2V24a2 2 0 0 12-2h18.437c9.083 0 ↴
18             ↴ 15.044 4.92 15.044 12.474 0 5.302-4.01 10.049-9.119 10.88v.277C75.2
19             ↴ .31 7 46.394 80 51.21 80 57.863zM60.521 28.34H49.948v14.934h8.905z
20             ↴ c6.884 0 10.68-2.772 10.68-7.727 0-4.643-3.264-7.207-9.012-7.207z
21             ↴ zM49.948 49.2v16.458H60.91c7.167 0 10.964-2.876 10.964-8.281z
22             ↴ 0-5.406-3.903-8.178-11.425-8.178H49.948z" />

```

```

7   </path>
8   </symbol>
9   <symbol id="home" viewBox="0 0 16 16">
10  <path d="M8.354 1.146a.5.5 0 0 0-.708 0l-6 6A.5.5 0 0 0 1.5 7.5v7a.5.5 0
    ↴ 0 0 .5.5h4.5a.5.5 0 0 0 -.5-.5v-4h2v4a.5.5 0 0 0 .5.5H14a.5.5 0 0 0
    ↴ .5-.5v-7a.5.5 0 0 0 -.14 6-.354L13 5.793V2.5a.5.5 0 0 0 -.5-.5h-1a2
    ↴ .5.5 0 0 0 -.5.5v1.293L8.354 1.146zM2.5 14V7.707l5.5-5.5 5.5 5.5
    ↴ V14H10v-4a.5.5 0 0 0 -.5-.5h-3a.5.5 0 0 0 -.5.5v4H2.5z" />
11  </symbol>
12  <symbol id="speedometer2" viewBox="0 0 16 16">
13  <path d="M8 4a.5.5 0 0 1 .5.5V6a.5.5 0 0 1-1 0V4.5A.5.5 0 0 1 8 4zM3.732
    ↴ 5.732a.5.5 0 0 1 .707 0l.915.914a.5.5 0 1 1-.708.708l-.914-.915a2
    ↴ .5.5 0 0 1 0-.707zM2 10a.5 .5 0 0 1 .5-.5h1.586a.5.5 0 0 1 0 1H2.5
    ↴ A.5.5 0 0 1 2 10zm9.5 0a.5.5 0 0 1 .5-.5h1.5a.5.5 0 0 1 0 1H12a2
    ↴ .5.5 0 0 1 -.5-.5zm.754-4.246a.389.389 0 0 0 -.527-.02L7.547 9.31a.
    ↴ 91.91 0 1 0 1.302 1.258l3.434-4.297a.389.389 0 0 0 -.029-.518z" />
14  <path fill-rule="evenodd" d="M0 10a8 8 0 1 1 15.547 2.661c-.442
    ↴ 1.253-1.845 1.602-2.932 1.25C11.309 13.488 9.475 13 8 13c-1.474
    ↴ 0-3.31.488-4.615.911-1.087.352-2.49 .003-2.932-1.25A7.988 7.988 0
    ↴ 0 1 0 10zm8-7a7 7 0 0 0-6.603 9.329c.203.575.923.876 1.68.63C4.397
    ↴ 12.533 6.358 12 8 12s3.604.532 4.923.96c.757.245 1.477-.056
    ↴ 1.68-.631A 7 7 0 0 0 8 3z" />
15  </symbol>
16  <symbol id="table" viewBox="0 0 16 16">
17  <path d="M0 2a2 2 0 0 1 2-2h12a2 2 0 0 1 2 2v12a2 2 0 0 1-2 2H2a2 2 0 0
    ↴ 1-2-2V2zm15 2h-4v3h4V4zm0 4h-4v3h4V8zm0 4h-4v3h3a1 1 0 0 0 1-1v-2
    ↴ zm-5 3v-3H6v3h4zm-5 0v-3H1 v2a1 1 0 0 0 1 1h3zm-4-4h4V8H1v3zm0-4
    ↴ h4V4H1v3zm5-3v3h4V4H6zm4 4H6v3h4V8z" />
18  </symbol>
19  <symbol id="people-circle" viewBox="0 0 16 16">
20  <path d="M11 6a3 3 0 1 1-6 0 3 3 0 0 1 6 0z" />
21  <path fill-rule="evenodd" d="M0 8a8 8 0 1 1 16 0A8 8 0 0 1 0 8zm8-7a7 7 0
    ↴ 0 0-5.468 11.37C3.242 11.226 4.805 10 8 10s4.757 1.225 5.468 2.37
    ↴ A7 7 0 0 0 8 1z" />
22  </symbol>
23  <symbol id="grid" viewBox="0 0 16 16">
24  <path d="M1 2.5A1.5 1.5 0 0 1 2.5 1h3A1.5 1.5 0 0 1 7 2.5v3A1.5 1.5 0 0 1
    ↴ 5.5 7h-3A1.5 1.5 0 0 1 1 5.5v-3zM2.5 2a.5.5 0 0 0-5.5v3a.5.5 0 0
    ↴ 0 .5.5h3a.5.5 0 0 0 .5-.5v-3a.5.5 0 0 0-5.5h-3zm6.5.5A1.5 1.5
    ↴ 0 0 1 10.5 1h3A1.5 1.5 0 0 1 15 2.5v3A1.5 1.5 0 0 1 13.5 7h-3A1.5
    ↴ 1.5 0 0 1 9 5.5v-3zm1.5-.5a.5.5 0 0 0-5.5v3a.5.5 0 0 0 .5.5 h3a2
    ↴ .5.5 0 0 0 .5-.5v-3a.5.5 0 0 0-5.5h-3zM1 10.5A1.5 1.5 0 0 1 2.5
    ↴ 9h3A1.5 1.5 0 0 1 7 10.5v3A1.5 1.5 0 0 1 5.5 15h-3A1.5 1.5 0 0 1 1
    ↴ 13.5v-3zm1.5-.5a.5.5 0 0 0-5.5 v3a.5.5 0 0 0 .5.5h3a.5.5 0 0 0
    ↴ .5-.5v-3a.5.5 0 0 0-5.5h-3zm6.5.5A1.5 1.5 0 0 1 10.5 9h3a1.5 1.5
    ↴ 0 0 1 1.5 1.5v3a1.5 1.5 0 0 1-1.5 1.5h-3A1.5 1.5 0 0 1 9 13.5v-3
    ↴ zm1. 5-.5a.5.5 0 0 0-5.5v3a.5.5 0 0 0 .5.5h3a.5.5 0 0 0 .5-.5v-3a2
    ↴ .5.5 0 0 0-5.5h-3z" />
25  </symbol>
26  <symbol id="collection" viewBox="0 0 16 16">
27  <path d="M2.5 3.5a.5.5 0 0 1 0-1h11a.5.5 0 0 1 0 1h-11zm2-2a.5.5 0 0 1
    ↴ 0-1h7a.5.5 0 0 1 0 1h-7zM0 13a1.5 1.5 0 0 0 1.5 1.5h13A1.5 1.5 0 0
    ↴ 0 16 13V6a1.5 1.5 0 0 0-1 .5-1.5h-13A1.5 1.5 0 0 0 6v7zm1.5.5A2
    ↴ .5.5 0 0 1 1 13V6a.5.5 0 0 1 .5-.5h13a.5.5 0 0 1 .5.5v7a.5.5 0 0
    ↴ 1-.5.5h-13z" />
28  </symbol>
29  <symbol id="calendar3" viewBox="0 0 16 16">
```

```

30 <path d="M14 0H2a2 2 0 0 0-2 2v12a2 2 0 0 0 2 2h12a2 2 0 0 0 2-2V2a2 2 0 ↵
   ↵ 0 0-2-2zM1 3.857C1 3.384 1.448 3 2 3h12c.552 0 1 .384 1 .857v10 ↵
   ↵ .286c0 .473-.448.857-1 .857 H2c-.552 0-1-.384-1-.857V3.857z" />
31 <path d="M6.5 7a1 1 0 1 0 0-2 1 1 0 0 0 0 2zm3 0a1 1 0 1 0 0-2 1 1 0 0 0 ↵
   ↵ 0 2zm3 0a1 1 0 1 0 0-2 1 1 0 0 0 0 2zm-9 3a1 1 0 1 0 0-2 1 1 0 0 0 ↵
   ↵ 0 2zm3 0a1 1 0 1 0 0-2 1 1 0 0 0 0 2zm3 0a1 1 0 1 0 0-2 1 1 0 0 0 ↵
   ↵ 0 2zm3 0a1 1 0 1 0 0-2 1 1 0 0 0 0 2zm-9 3a1 1 0 1 0 0-2 1 1 0 0 0 ↵
   ↵ 0 0 2zm3 0a1 1 0 1 0 0-2 1 1 0 0 0 0 2zm3 0a1 1 0 1 0 0-2 1 1 0 0 0 ↵
   ↵ 0 0 2z" />
32 </symbol>
33 <symbol id="chat-quote-fill" viewBox="0 0 16 16">
34 <path d="M16 8c0 3.866-3.582 7-8 7a9.06 9.06 0 0 1-2.347-.306c ↵
   ↵ -.584.296-1.925.864-4.181 ↵
   ↵ 1.234-.2.032-.352-.176-.273-.362.354-.836.674-1.95.77-2.966C.744 ↵
   ↵ 11.37 0 9.76 0 8c0-3.866 3.582-7 8-7s8 3.134 8 7zM7.194 6.766a1 ↵
   ↵ .688 1.688 0 0 0-.227-.272 1.467 1.467 0 0 0-.469-.324l-.008-.004 ↵
   ↵ A1.785 1.785 0 0 0 5.734 6C4.776 6 4 6.746 4 7.66 7c0 .92.776 ↵
   ↵ 1.666 1.734 1.666.343 0 .662-.095.931-.26-.137.389-.39.804-.81 ↵
   ↵ 1.22a.405.405 0 0 0 .011.59c.173.16.447.155.614-.01 1.334-1.329 ↵
   ↵ 1.37-2.758.941-3.706a2.461 2 .461 0 0 0-.227-.4zM11 9.073c ↵
   ↵ -.136.389-.39.804-.81 1.22a.405.405 0 0 0 .012.59c ↵
   ↵ .172.16.446.155.613-.01 1.334-1.329 1.37-2.758.942-3.706a2.466 ↵
   ↵ 2.466 0 0 0-.228-.4 1.686 1.686 0 0 0-.227-.273 1.466 1.466 0 0 ↵
   ↵ 0-.469-.324l-.008-.004A1.785 1.785 0 0 0 10.07 6c-.957 ↵
   ↵ 0-1.734.746-1.734 1.667 0 .92.777 1.666 1.734 1.666.343 0 ↵
   ↵ .662-.095.931-.26z" />
35 </symbol>
36 <symbol id="cpu-fill" viewBox="0 0 16 16">
37 <path d="M6.5 6a.5.5 0 0 0-.5.5v3a.5.5 0 0 0 .5.5h3a.5.5 0 0 0 .5-.5v-3a ↵
   ↵ .5.5 0 0 0-.5-.5h-3z" />
38 <path d="M5.5.5a.5.5 0 0 0-1 0V2A2.5 2.5 0 0 0 2 4.5H.5a.5.5 0 0 0 0 1 ↵
   ↵ H2v1H.5a.5.5 0 0 0 1H2v1H.5a.5.5 0 0 0 0 1H2v1H.5a.5.5 0 0 0 0 1 ↵
   ↵ H2A2.5 2.5 0 0 0 4.5 14v1.5 a.5.5 0 0 0 1 0V14h1v1.5a.5.5 0 0 0 1 ↵
   ↵ 0V14h1v1.5a.5.5 0 0 0 1 0V14h1v1.5a.5.5 0 0 0 1 0V14a2.5 2.5 0 0 0 2 ↵
   ↵ 2.5-2.5h1.5a.5.5 0 0 0 0-1H14v-1h1.5a.5.5 0 0 0 0-1H14A2.5 2.5 0 0 0 11.5 2V.5 ↵
   ↵ .5.5 0 0 0 0-1H14v-1h1.5a.5.5 0 0 0 0-1H14A2.5 2.5 0 0 0 11.5 2V.5 ↵
   ↵ a.5.5 0 0 0-1 0V2h-1V.5a.5.5 0 0 0-1 0V2h-1V.5a.5.5 0 0 0-1 0V2h-1 ↵
   ↵ V.5zm1 4.5h3A1.5 1.5 0 0 1 11.6.5v3 A1.5 1.5 0 0 1 9.5 11h-3A1.5 ↵
   ↵ 1.5 0 0 1 5 9.5v-3A1.5 1.5 0 0 1 6.5 5z" />
39 </symbol>
40 <symbol id="gear-fill" viewBox="0 0 16 16">
41 <path d="M9.405 1.05c-.413-1.4-2.397-1.4-2.81 0l-.1.34a1.464 1.464 0 0 ↵
   ↵ 1-2.105.872l-.31-.17c-1.283-.698-2.686.705-1.987 1.987l.169.311c ↵
   ↵ .446.82.023 1.841-.872 2.105 l-.34.1c-1.4.413-1.4 2.397 0 2.81l ↵
   ↵ .34.1a1.464 1.464 0 0 1 .872 2.105l-.17.31c-.698 1.283.705 2.686 ↵
   ↵ 1.987 1.987l.311-.169a1.464 1.464 0 0 1 2.105.872l.1.34c.413 1.4 ↵
   ↵ 2.3 97 1.4 2.81 0l.1-.34a1.464 1.464 0 0 1 2.105-.872l.31.17c1 ↵
   ↵ .283.698 2.686-.705 1.987-1.987l-.169-.311a1.464 1.464 0 0 1 ↵
   ↵ .872-2.105l.34-.1c1.4-.413 1.4-2.397 0-2.81l-.34 -.1a1.464 1.464 0 ↵
   ↵ 0 1-.872-2.105l.17-.31c.698-1.283-.705-2.686-1.987-1.987l ↵
   ↵ -.311.169a1.464 1.464 0 0 1-2.105-.872l-.1-.34zM8 10.93a2.929 ↵
   ↵ 2.929 0 1 1 0-5.86 2.929 2.929 0 0 1 0 5.858z" />
42 </symbol>
43 <symbol id="speedometer" viewBox="0 0 16 16">
44 <path d="M8 2a.5.5 0 0 1 .5.5V4a.5.5 0 0 1-1 0V2.5A.5.5 0 0 1 8 2zM3.732 ↵
   ↵ 3.732a.5.5 0 0 1 .707 0l.915.914a.5.5 0 1 1-.708.708l-.914-.915a ↵
   ↵ .5.5 0 0 1 0-.707zM2 8a.5.5 0 0 1 .5-.5h1.586a.5.5 0 0 1 0 1H2.5A ↵

```

```

    ↴ .5.5 0 0 1 2 8zm9.5 0a.5.5 0 0 1 .5-.5h1.5a.5.5 0 0 1 0 1H12a.5.5 ↵
    ↴ 0 0 1-.-.5zm.754-4.246a.389.389 0 0 0-.527-.02L7.547 7.31A.91 .91 ↵
    ↴ 0 1 0 8.85 8.569l3.434-4.297a.389.389 0 0 0-.029-.518z" />
45 <path fill-rule="evenodd" d="M6.664 15.889A8 8 0 1 1 9.336.11a8 8 0 0 ↵
    ↴ 1-2.672 15.78zm-4.665-4.283A11.945 11.945 0 0 1 8 10c2.186 0 ↵
    ↴ 4.236.585 6.001 1.606a7 7 0 1 0- 12.002 0z" />
46 </symbol>
47 <symbol id="toggles2" viewBox="0 0 16 16">
48 <path d="M9.465 10H12a2 2 0 1 1 0 4H9.465c.34-.588.535-1.271.535-2 ↵
    ↴ 0-.729-.195-1.412-.535-2z" />
49 <path d="M6 15a3 3 0 1 0 0-6 3 3 0 0 0 0 6zm0 1a4 4 0 1 1 0-8 4 4 0 0 1 0 ↵
    ↴ 8zm.535-10a3.975 3.975 0 0 1-.409-1H4a1 1 0 0 1 0-2h2.126c ↵
    ↴ .091-.355.23-.69.41-1H4a2 2 0 1 0 0 4h2.535z" />
50 <path d="M14 4a4 4 0 1 1-8 0 4 4 0 0 1 8 0z" />
51 </symbol>
52 <symbol id="tools" viewBox="0 0 16 16">
53 <path d="M1 0L0 1l2.2 3.081a1 1 0 0 0 .815.419h.07a1 1 0 0 1 .708.293l2 ↵
    ↴ .675 2.675-2.617 2.654A3.003 3.003 0 0 0 13a3 3 0 1 0 5.878-.851 ↵
    ↴ l2.654-2.617.968.968-.305. 914a1 1 0 0 0 .242 1.023l3.356 3.356a1 ↵
    ↴ 1 0 0 0 1.414 0l1.586-1.586a1 1 0 0 0 0-1.414l-3.356-3.356a1 1 0 0 ↵
    ↴ 0-1.023-.242L10.5 9.5l-.96-.96 2.68-2.643A3.005 3.005 0 0 0 1 6 3 ↵
    ↴ c0-.269-.035-.53-.102-.777l-2.14 2.141L12 4l-.364-1.757L13.777.102 ↵
    ↴ a3 3 0 0 0-3.675 3.68L7.462 6.46 4.793 3.793a1 1 0 0 1 -.293-.707v ↵
    ↴ -.071a1 1 0 0 0-.419-.814L1 0zm9. 646 10.646a.5.5 0 0 1 .708 0l3 3 ↵
    ↴ a.5.5 0 0 1-.708.708l-3-3a.5.5 0 0 1 0-.708zM3 11l ↵
    ↴ .471.242.529.026.287.445.445.287.026.529L5 13l ↵
    ↴ -.242.471-.026.529-.445.287-.287.445-.5 29.026L3 15l-.471-.242L2 ↵
    ↴ 14.732l-.287-.445L1.268 14l-.026-.529L1 13l ↵
    ↴ .242-.471.026-.529.445-.287.287-.445.529-.026L3 11z" />
54 </symbol>
55 <symbol id="chevron-right" viewBox="0 0 16 16">
56 <path fill-rule="evenodd" d="M4.646 1.646a.5.5 0 0 1 .708 0l6 6a.5.5 0 0 ↵
    ↴ 1 0 .708l-6 6a.5.5 0 0 1-.708-.708L10.293 8 4.646 2.354a.5.5 0 0 1 ↵
    ↴ 0-.708z" />
57 </symbol>
58 <symbol id="geo-fill" viewBox="0 0 16 16">
59 <path fill-rule="evenodd" d="M4 4a4 4 0 1 1 4.5 3.969V13.5a.5.5 0 0 1-1 0 ↵
    ↴ V7.97A4 4 0 0 1 4 3.999zm2.493 8.574a.5.5 0 0 1-.411.575c ↵
    ↴ -.712.118-1.28.295-1.655.493a1.31 9 1.319 0 0 0-.37.265.301.301 0 ↵
    ↴ 0 0-.057.09V14l.002.008a.147.147 0 0 0 .016.033.617.617 0 0 0 ↵
    ↴ .145.15c.165.13.435.27.813.395.751.25 1.82.414 3.024.414s2 ↵
    ↴ .273-.163 3.024 -.414c.378-.126.648-.265.813-.395a.619.619 0 0 0 ↵
    ↴ .146-.15.148.148 0 0 0 .015-.033L12 14v-.004a.301.301 0 0 0 ↵
    ↴ 0-.057-.09 1.318 1.318 0 0 0-.37-.264c-.376-.198-.943-.375-1 ↵
    ↴ .655-.493a.5.5 0 1 1 .164-.986c.77.127 1.452.328 1.957.594C12.5 13 ↵
    ↴ 13 13.4 13 14c0 ↵
    ↴ .426-.26.752-.544.977-.29.228-.68.413-1.116.558-.878.293-2.059.465-3.34.465- ↵
    ↴ 0-2.462-.172-3.34-.465-.436-.145-.826-.33-1.116-.558C3.26 14.752 ↵
    ↴ 3 14.426 3 14c0-.599.5-1 .961-1.243.505-.266 1.187-.467 ↵
    ↴ 1.957-.594a.5.5 0 0 1 .575.411z" />
60 </symbol>
61 </svg>
62 <div class="d-flex flex-column flex-shrink-0 p-3 bg-light" ↵
63   style="width: 280px">
64   <a href="#" ↵
65     class="d-flex align-items-center mb-3 mb-md-0 me-md-auto link-dark" ↵
       text-decoration-none">
```

```
66      
67      <span class="fs-4">Registrar</span>
68  </a>
69  <hr>
70  <ul class="nav nav-pills flex-column mb-auto">
71    <li class="nav-item">
72      <a href="" class="nav-link active" aria-current="page">
73        <svg class="bi me-2" width="16" height="16">
74          <use xlink:href="#home" />
75        </svg>
76        Home
77      </a>
78    </li>
79    <li>
80      <a href="/admin" class="nav-link link-dark">
81        <svg class="bi me-2" width="16" height="16">
82          <use xlink:href="#speedometer2" />
83        </svg>
84        Add User
85      </a>
86    </li>
87    <li>
88      <a href="#" class="nav-link link-dark">
89        <svg class="bi me-2" width="16" height="16">
90          <use xlink:href="#table" />
91        </svg>
92        Orders
93      </a>
94    </li>
95    <li>
96      <a href="#" class="nav-link link-dark">
97        <svg class="bi me-2" width="16" height="16">
98          <use xlink:href="#grid" />
99        </svg>
100       Products
101     </a>
102   </li>
103   <li>
104     <a href="#" class="nav-link link-dark">
105       <svg class="bi me-2" width="16" height="16">
106         <use xlink:href="#people-circle" />
107       </svg>
108       Customers
109     </a>
110   </li>
111 </ul>
112 <hr>
113 <div class="dropdown">
114   {% if user.is_authenticated %}
115     <a class="nav-item nav-link" href="{% url 'logout' %}">
116       Log out <i class="fas fa-sign-out-alt"></i>
117     </a>
118   {% else %}
119     <a class="nav-item nav-link" href="{% url 'login' %}">
120       Log in <i class="fas fa-sign-in-alt"></i>
121     </a>
```

```
122     {% endif %}  
123     </div>  
124 </div>
```

registrar/static/registrar/navbar.css

```
1 .row-container {  
2     display: flex;  
3     width: 100%;  
4     height: 100%;  
5     flex-direction: column;  
6     background-color: #DCDCDC;  
7     overflow: hidden;  
8 }  
9 .first-row {  
10    background-color: lime;  
11 }  
12 .second-row {  
13     flex-grow: 1;  
14     border: none;  
15     margin: 0;  
16     padding: 0;  
17 }  
18 .bd-placeholder-img {  
19     font-size: 1.125rem;  
20     text-anchor: middle;  
21     -webkit-user-select: none;  
22     -moz-user-select: none;  
23     user-select: none;  
24 }  
25 @media (min-width: 768px) {  
26     .bd-placeholder-img-lg {  
27         font-size: 3.5rem;  
28     }  
29 }  
30 body {  
31     min-height: 100vh;  
32     min-height: -webkit-fill-available;  
33 }  
34 html {  
35     height: -webkit-fill-available;  
36 }  
37 main {  
38     display: flex;  
39     flex-wrap: nowrap;  
40     height: 100vh;  
41     height: -webkit-fill-available;  
42     max-height: 100vh;  
43     overflow-x: auto;  
44     overflow-y: hidden;  
45 }  
46 .b-example-divider {  
47     flex-shrink: 0;  
48     width: 1.5rem;  
49     height: 100vh;
```

```
50    background-color: rgba(0, 0, 0, .1);
51    border: solid rgba(0, 0, 0, .15);
52    border-width: 1px 0;
53    box-shadow: inset 0 .5em 1.5em rgba(0, 0, 0, .1), inset 0 .125em .5em ↴
54      ↴ rgba(0, 0, 0, .15);
55  }
56  .bi {
57    vertical-align: -.125em;
58    pointer-events: none;
59    fill: currentColor;
60  }
61  .dropdown-toggle {
62    outline: 0;
63  }
64  .nav-flush .nav-link {
65    border-radius: 0;
66  }
67  .btn-toggle {
68    display: inline-flex;
69    align-items: center;
70    padding: .25rem .5rem;
71    font-weight: 600;
72    color: rgba(0, 0, 0, .65);
73    background-color: transparent;
74    border: 0;
75  }
76  .btn-toggle:hover, .btn-toggle:focus {
77    color: rgba(0, 0, 0, .85);
78    background-color: #d2f4ea;
79  }
80  .btn-toggle::before {
81    width: 1.25em;
82    line-height: 0;
83    content: url("data:image/svg+xml,%3csvg xmlns='http://www.w3.org/2000/↗
84      ↴ svg' width='16' height='16' viewBox='0 0 16 16'%3e%3cpath fill='none' ↴
85      ↴ stroke='rgba%280,0,0,.5%29' stroke-linecap='round' ↴
86      ↴ stroke-linejoin='round' stroke-width='2' d='M5 14l6-6-6-6'%3e%3c/svg%3e");
87    transition: transform .35s ease;
88    transform-origin: .5em 50%;
89  }
90  .btn-toggle[aria-expanded="true"] {
91    color: rgba(0, 0, 0, .85);
92  }
93  .btn-toggle[aria-expanded="true"]::before {
94    transform: rotate(90deg);
95  }
96  .btn-toggle-nav a {
97    display: inline-flex;
98    padding: .1875rem .5rem;
99    margin-top: .125rem;
100   margin-left: 1.25rem;
101   text-decoration: none;
102  }
103  .btn-toggle-nav a:hover, .btn-toggle-nav a:focus {
104    background-color: #d2f4ea;
105  }
```

```

102 .scrollarea {
103     overflow-y: auto;
104 }
105 .fw-semibold {
106     font-weight: 600;
107 }
108 .lh-tight {
109     line-height: 1.25;
110 }

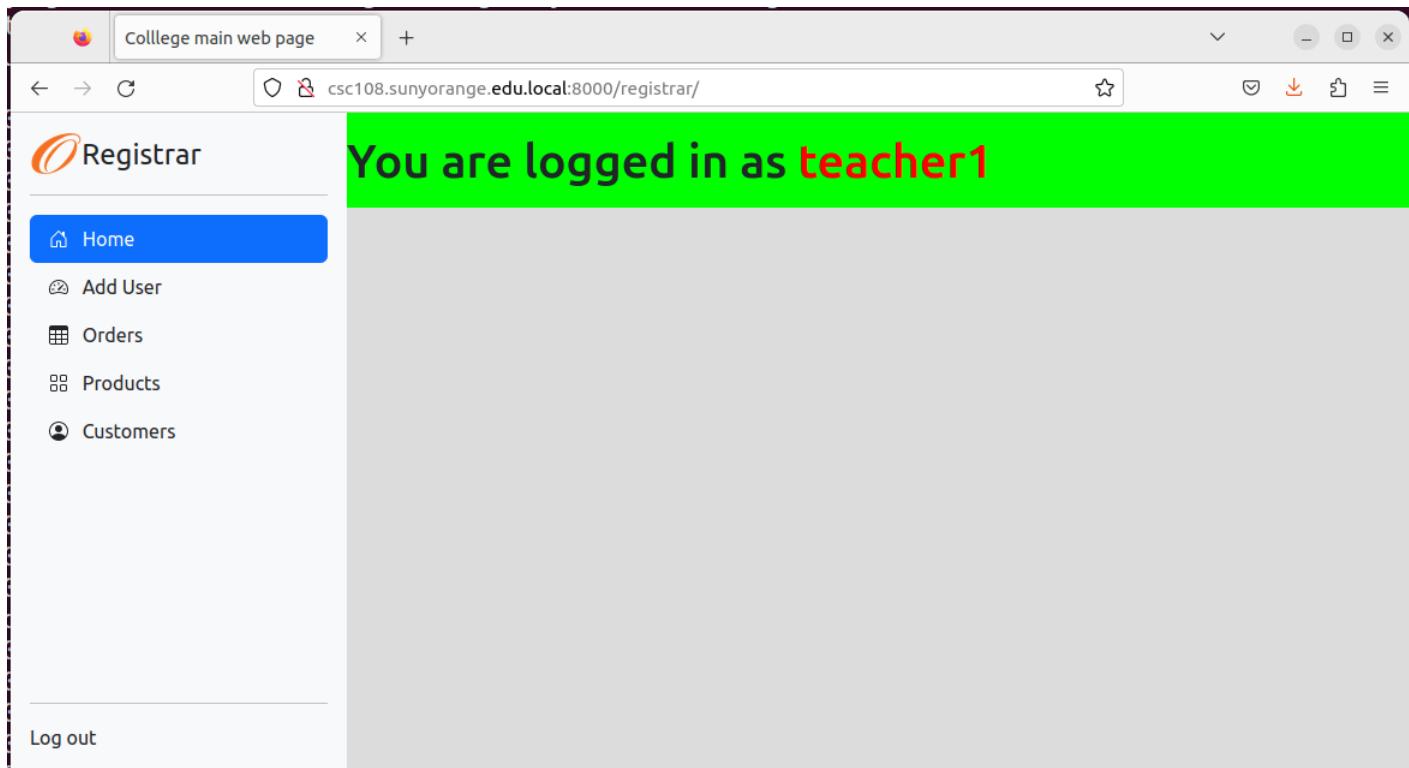
```

registrar/static/registrar/navbar.js

```

1 /* global bootstrap: false */
2 (function () {
3     'use strict'
4     var tooltipTriggerList = [].slice.call(document.querySelectorAll('[data-'
5         + 'bs-toggle="tooltip"]'))
6     tooltipTriggerList.forEach(function (tooltipTriggerEl) {
7         new bootstrap.Tooltip(tooltipTriggerEl)
8     })
9 })()

```

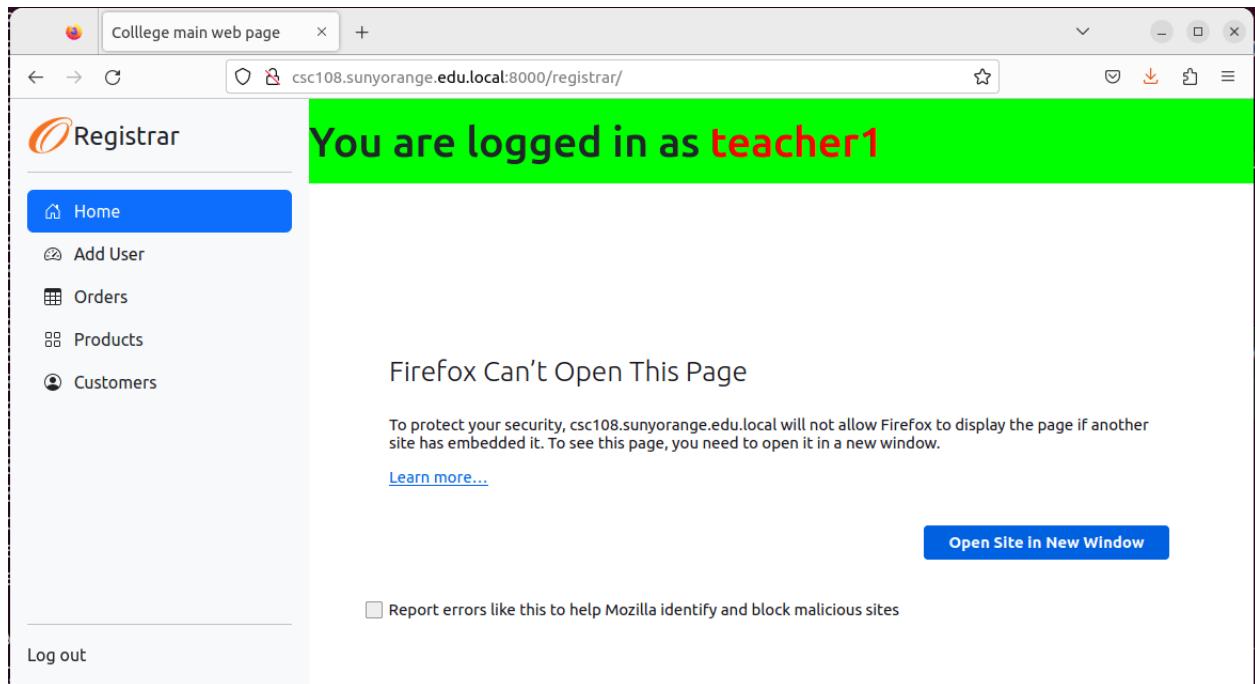


Sidebar based on example from:

<https://getbootstrap.com/docs/5.0/examples/sidebars/#>

12.2.2 X-frame-options

The java script on our navbar will open up a new i-frame and load the page into it. By default this is not allowed to prevent loading clickjacking or loading javascript from unknown sources. We can modify it to allow loading a new page, but only from our domain.



Log out Report errors like this to help Mozilla identify and block malicious sites

Filter Output Errors Warnings Logs Info Debug CSS XHR Requests Top ↕

Console

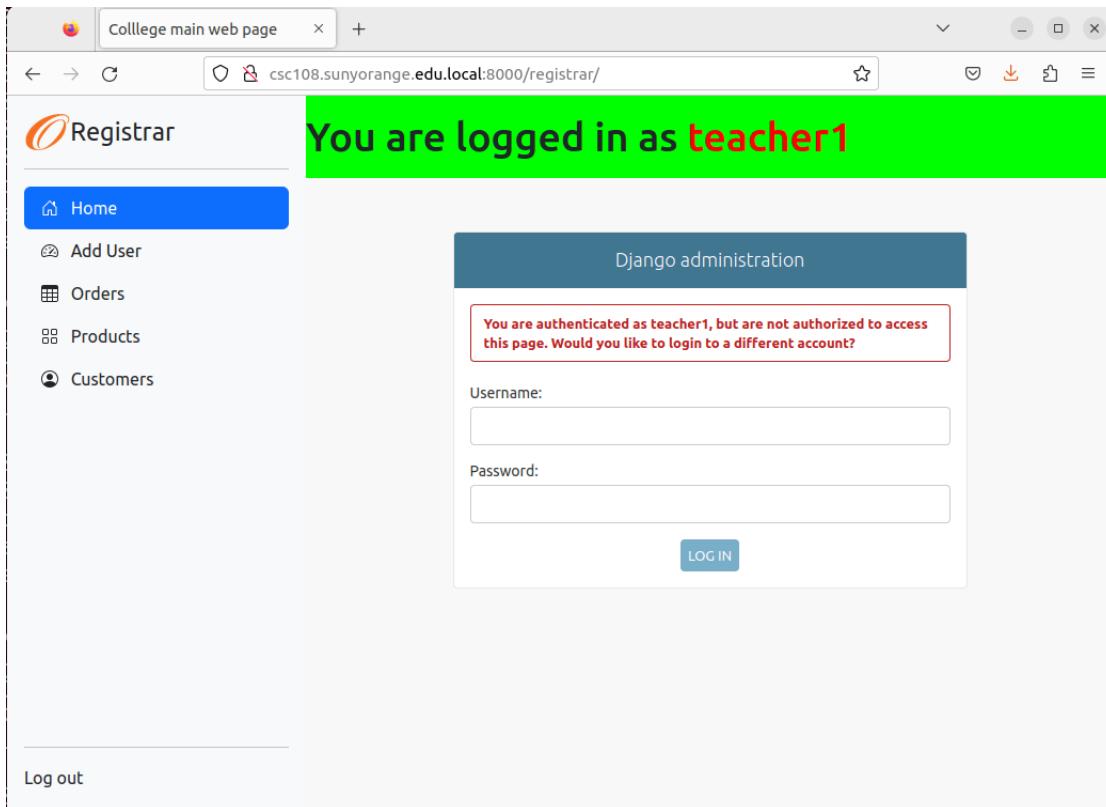
```
⠄ The loading of "http://csc108.sunyorange.edu.local:8000/admin/login/?next=/admin/" in a frame is denied by "X-Frame-Options" directive set to "DENY". \[Learn More\]
⠄ The loading of "http://csc108.sunyorange.edu.local:8000/admin/login/?next=/admin/" in a frame is denied by "X-Frame-Options" directive set to "DENY". \[Learn More\]
```

<https://docs.djangoproject.com/en/4.2/ref/clickjacking/>

[mozilla x-frame-options](#)

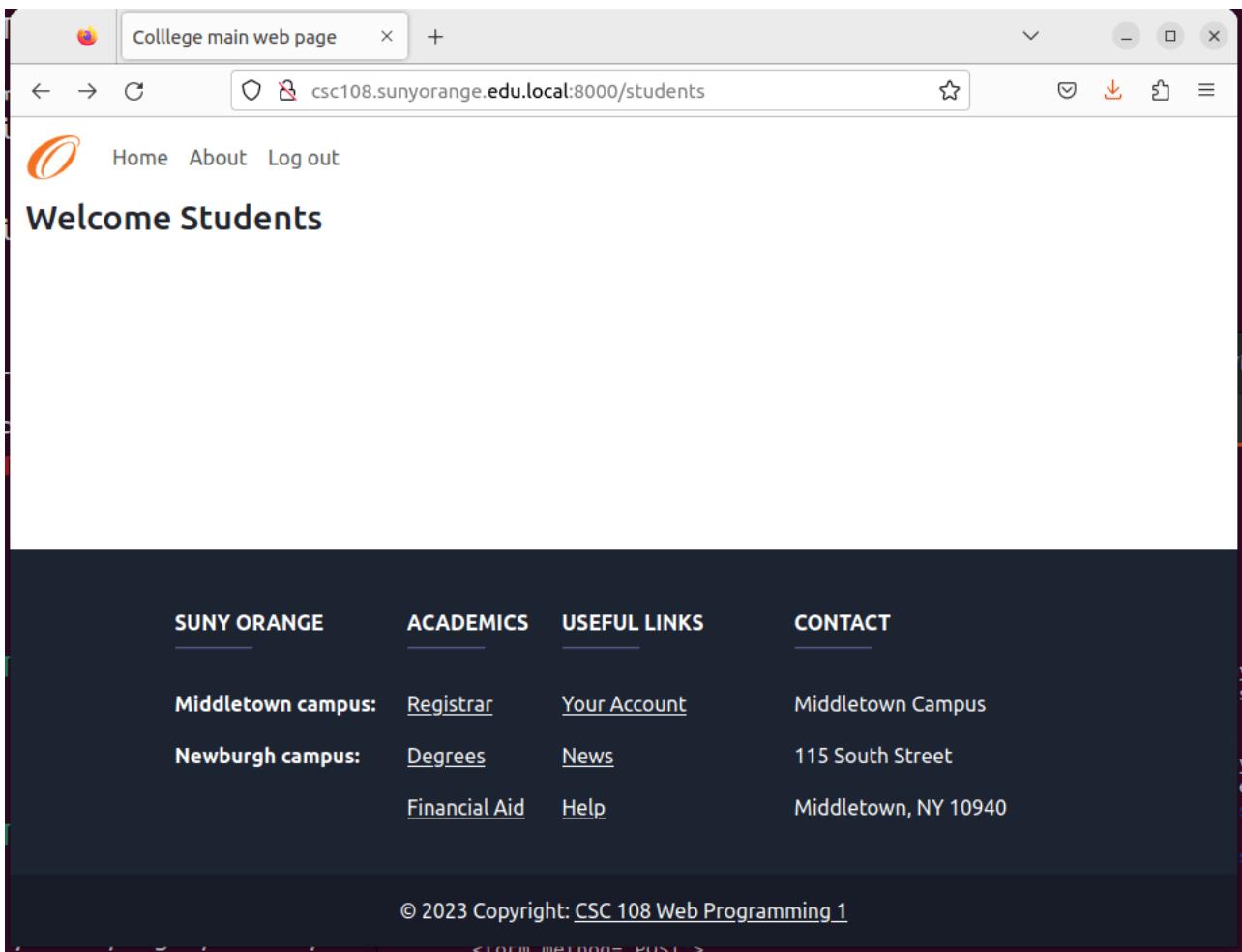
X_FRAME_OPTIONS = "SAMEORIGIN"

Add the above to our main settings.py file and reload the server.

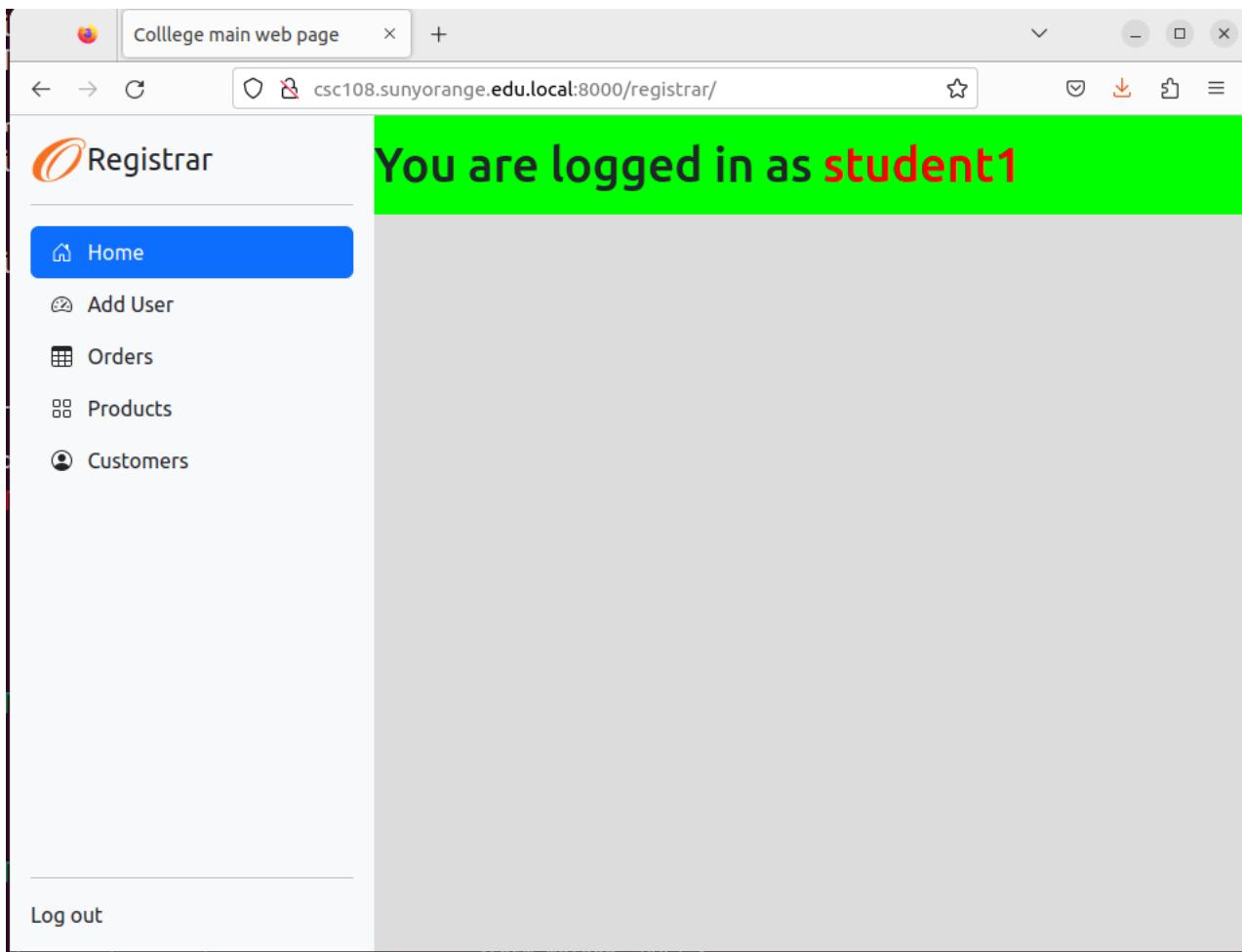


12.3 ➤ revisit our login page

At this time student1 is a student member and should not have access to the registrar page.



By changing the URL to the registrar page we can log in.



12.3.1 Authentication vs Authorization

Authentication and authorization are two critical concepts in the realm of information security, each with a unique purpose and role. While they are related and often work together, they are not interchangeable. Here's how they differ:

Authentication

Authentication verifies who a user is, or at least that they are who they say they are. It involves confirming the identity of a person or system. This process typically involves credentials like usernames, passwords, biometric data, or multi-factor authentication. A straightforward example would be logging into your email account; you enter your username and password, and if they are correct, you are authenticated and allowed to access your account.

Authorization

Authorization, on the other hand, happens after authentication and determines what a user is allowed to do or what resources they are allowed to access. For example, a user might be authenticated to log into a system, but they are only authorized to access certain parts of that system based on their user privileges.

Comparing these processes to a real-world example, when you go through security in an airport, you show your ID to authenticate your identity. Then, when you arrive at the gate, you present your boarding pass to the flight attendant, so they can authorize you to board your flight and allow access to the plane.

In conclusion, while authentication and authorization are often used together in the context of access control, they represent distinct stages in the process. Authentication is about verifying identity, while authorization is about granting access based on that verified identity.

AUTHENTICATION	AUTHORIZATION
Usually the first step of a security access control	Usually comes after authentication
Verifies the user's identity	Grants or denies permissions to the user to do something
Common methods include: username, password, answer to security question, code send via SMS or Email	permissions are granted and monitored by the organization
Can use biometric data like fingerprint, face recognition, retinal scan	Common methods include: role-based access control and attribute-based access control
It's visible by the user	It's not visible to the user
It's changeable by the user (in some cases)	Cannot be changed by the user

12.3.2 Make changes to user login view.py

```

1 if user is not None:
2     login(request, user)
3     messages.success(request, f"Hello {user.username}! You have been ↴
4         ↴ logged in")
5     queryset = CustomUser.objects.filter(username=form.cleaned_data['↳
6         ↴ username']).values('status')
7     user_type = queryset[0]['status']
8     request.session['role'] = user_type ##### add role to session
9     #print(user_type)
10    if(user_type == 'faculty'):
11        return redirect('faculty')
12    elif(user_type == 'admin'):
13        return redirect('registrar\')
14    elif(user_type == 'student'):
15        return redirect('students')
16    else:
17        return redirect('/')

```

request.session['role'] = user_type ##### add role to session

We can modify the items in the session by adding a new key into the **request.session** dictionary.

12.3.3

Use the role to restrict users from accessing pages that their role does not permit

Edit the registrar view.py

```

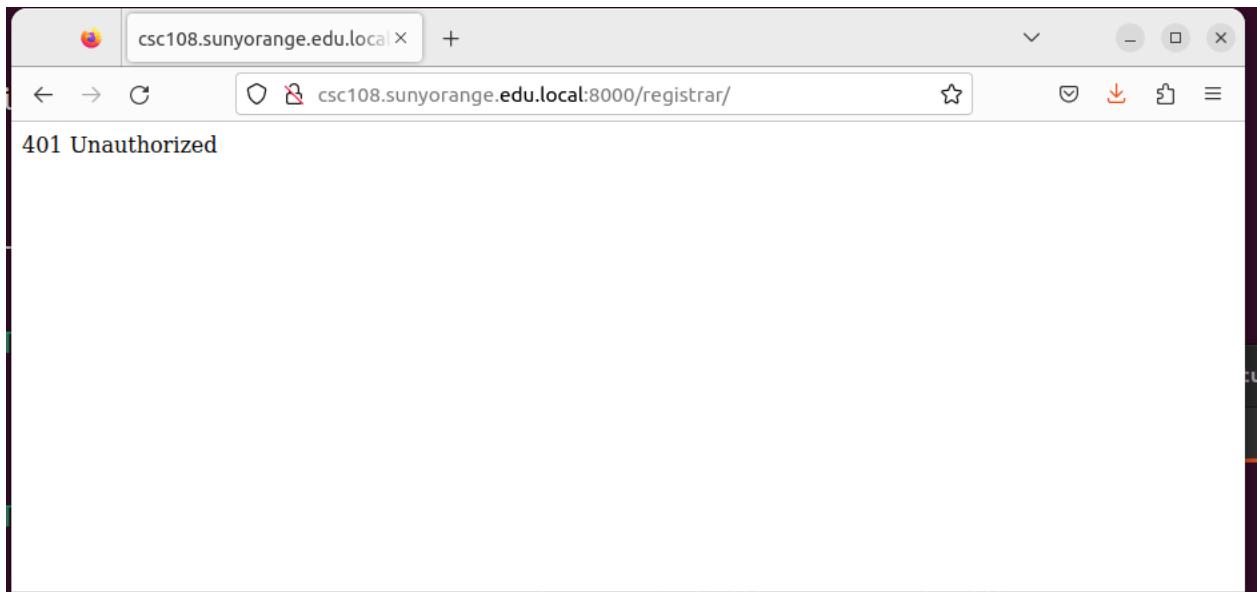
new registrar view.py

1 @login_required
2 def home(request):
3     if(request.session['role'] != 'admin'):
4         return HttpResponseRedirect('401 Unauthorized', status=401)
5     else:
6         return render(request,"registrar/home.html")

```

When we log in as student and try to change the url to registrar we get a

different page.



However if a an admin account logs in we will ge the correct response.

we can also do a redirect

12.4 ➤ create a page to add a user

What are the required fields for a user? The nice thing about django is that we can create a form based on what is in the table without actually writing the form yourself.

registrar/forms.py

We can create a new form in pour forms.py or reuse an existing one.

registrar.forms.py

```
1 from django import forms
2 from django.contrib.auth.forms import UserCreationForm
3 from django.contrib.auth import get_user_model
4 from users.models import CustomUser
5
6 class AddUserForm(UserCreationForm):
7
8     class Meta:
```

```

9     model = CustomUser
10    fields = '__all__'

```

In order for the form to automatically know which field to add to the form it has to know which database model to use. In this case we used the user CustomUser model. To remind us of the model here is the code again.

```

from django.contrib.auth.models import AbstractUser
from django.db import models

class CustomUser(AbstractUser):

    STATUS = (
        ('student', 'student'),
        ('faculty', 'faculty'),
        ('admin', 'admin'),
    )

    email = models.EmailField(unique=True)
    status = models.CharField(max_length=100, choices=STATUS, default='regular')
    description = models.TextField("Description", max_length=600, default='', blank=True)
    def __str__(self):
        return self.username

```

```
from users.models import CustomUser
```

the users. part is the name of the app where the model is stored. we can take a model from any app and use it in any other app by just specifying the app name.

```
class Meta:
```

The **class Meta** in Django is a special class that is used to contain any configurations or settings for the model it is associated with. The Meta class allows you to specify a number of different settings or behaviors about the model, including (but not limited to):

- **verbose_name** and **verbose_name_plural**: These provide human-readable names for the model, which are used in Django's automatically-generated admin site.
- **ordering**: This specifies the default ordering for objects of this model when you retrieve them from the database.

- **db_table:** The name of the database table to use for the model. If this isn't specified, Django will automatically derive a database table name from the name of the model's class.
- **abstract:** If True, this indicates that the model should be treated as an abstract base class. Abstract base classes are a way of declaring common fields that are inherited by the subclasses.
- **unique_together:** This is a list of lists of fields that must be unique when considered together.
- **index_together:** This is a list of lists of fields that should be indexed together.
- **get_latest_by:** Specifies the field to be used as a timestamp when calling the model's get_latest method.

For a full list see:

<https://docs.djangoproject.com/en/4.2/ref/models/options/>

The meta class tag can be used in other classes like forms not just the model.py class.

For forms refer to:

<https://docs.djangoproject.com/en/4.2/topics/forms/modelforms/>

12.4.1 add user creation form to our urls.py

registrar.urls.py

```
1 from django.urls import path
2
3 from . import views
4
5 urlpatterns = [
6     path("registrar/", views.home, name="registrar"),
7     path("registrar/adduser/", views.adduser, name="adduser"),
8 ]
9 ]
```

registrar/views.py

registrar.urls.py

```

1  from django.shortcuts import render
2  from django.http import HttpResponseRedirect
3  from django.template import loader
4  from django.contrib.auth.decorators import login_required
5
6  from .forms import AddUserForm
7  from django.contrib import messages
8
9  def index(request):
10     """ main landing page for our website """
11     return render(request, "registrar/home.html")
12 @login_required
13 def home(request):
14     if(request.session['role'] != 'admin'):
15         return HttpResponseRedirect('401 Unauthorized', status=401)
16         return render(request, '404.html', status=404)
17     else:
18         return render(request, "registrar/home.html")
19
20 @login_required
21 def adduser(request):
22     if(request.session['role'] != 'admin'):
23         return HttpResponseRedirect('401 Unauthorized', status=401)
24         return render(request, '404.html', status=404)
25     else:
26         if request.POST: #submitting a completed form
27             form = AddUserForm(request.POST)
28             if form.is_valid():
29                 form.save()
30                 return redirect('registrar')
31             else:
32                 for error in list(form.errors.values()):
33                     messages.error(request, error)
34         else: #not a post but a get. getting the form for first time
35             form = AddUserForm()
36             context = {}
37             context['form'] = form
38
39         return render(request, "registrar/adduser.html", context)

```

Added some new imports into our views.py

```

from .forms import AddUserForm
from django.contrib import messages

```

registrar/base2.html

registrar.templates/registrar/base2.html

```

1  {% load static %}
2  <!DOCTYPE html>
3  <html lang="en-US">
4      <head>
5          <meta charset="utf-8" >
6          <title>College main web page</title>
7          <link rel="stylesheet" type="text/css" href="{% static 'csc108/bootstrap' %}>
8              <-- 5.3.0-dist/css/bootstrap.min.css' %}">
9          <link rel="stylesheet" type="text/css" href="{% static 'registrar/navbar' %}>
10             <-- .css' %}">
11
12      </head>
13      <body>
14          <main>
15              <div style="width: 100%;>
16                  {% include 'csc108/includes/messaging.html' %}>
17                  {% block content %}>
18                  {% endblock %}>
19          </div>
20      </main>
21
22      <!-- Optional Javascript -->
23          <script src="{% static 'csc108/jquery3.7.0/jquery-3.7.0.min.js' %}"><-->
24              <-- /script>
25          <script src="{% static 'csc108/bootstrap-5.3.0-dist/js/bootstrap' %}>
26              <-- bundle.min.js' %}"></script>
27          <script src="{% static 'registrar/navbar.js' %}"></script>
28
29      <script>
30          $(document).ready(function(){
31              window.setTimeout(function() {
32                  $(".alert").fadeTo(500, 0).slideUp(500, function(){
33                      $(this).remove();
34                  });
35                  }, 5000);
36          });
37
38          $(function () {
39              $("li a").on("click", function () {
40                  var url = $(this).attr("href");
41                  $("#dvDisplay").html('<iframe class="row-container" >
42                      <-- frameborder="1" src="' + url + '" />');
43                  return false;
44              });
45          });
46
47
48      </body>
49  </html>

```

The only thing changed to the new base2.html was the removal of the sidebar.

registrar.templates/registrar/adduser.html

```
1  {% extends "registrar/base2.html" %}  
2  {% load static %}  
3  
4  {% block content %}  
5  <form action = "" method = "post">  
6    {% csrf_token %}  
7    {{ form.as_p }}  
8    <input type="submit" value=Submit">  
9  </form>  
10  
11  {% endblock content %}
```

The screenshot shows a Firefox browser window with the title bar "College main web page". The address bar displays the URL "csc108.sunyorange.edu.local:8000/registrar/". The page content is a user creation form for a "Registrar" system.

Left sidebar (Navigation):

- Home (selected)
- Add User
- Orders
- Products
- Customers

Main Content Area:

You are logged in as **admin**

User Details:

Password:

Last login:

Superuser status: Designates that this user has all permissions

Groups: The groups this user belongs to. A user will get all permissions for each group listed here.

User permissions:
admin | log entry | Can add log entry
admin | log entry | Can change log entry
admin | log entry | Can delete log entry
admin | log entry | Can view log entry

Username: Required. 150 characters

First name:

Last name:

Staff status: Designates whether the user can log into this admin site

Active: Designates whether this user should be treated as active

Date joined: 2023-07-06 17:29:40

Email:

Status:

Bottom Left:

Log out

Our form is displayed, but it is too large for our page. We will need to add scroll bars and change the order of items.

registrar/templates/registrar/base2.html add scroll bars

```
1 <main>
2     <div style="width: 100%; white-space: nowrap; overflow-x: scroll; ↴
3         ↴ overflow-y: scroll;">
4         {% include 'csc108/includes/messaging.html' %}
5         {% block content %}
6         {% endblock %}
7
8     </div>
9 </main>
```

12.5 ➤ crispy forms

Before we edit the fields we want to display lets make this form bit more presentable. By default there is no css or styling of any kind added to the django forms. One of the most popular way to make forms look better is by using a plugin called crispy forms.

<https://djangopackages.org/packages/p/django-crispy-forms/>

<https://github.com/django-crispy-forms/django-crispy-forms>

<https://django-crispy-forms.readthedocs.io/en/latest/index.html>

install crispy forms and crispy-bootstrap5

```
pip install django-crispy-forms
pip install crispy-bootstrap5
```

Make sure you have your env activated first.

```
(orange) occc@occc-VirtualBox:/csc108/project1/mywebsite$ ls
admin  csc108  db.sqlite3  faculty  manage.py  mywebsite  occc.txt  registrar  students  users
(orange) occc@occc-VirtualBox:/csc108/project1/mywebsite$ pip install django-crispy-forms
Collecting django-crispy-forms
  Downloading django_crispy_forms-2.0-py3-none-any.whl (31 kB)
Requirement already satisfied: django>=3.2 in /csc108/project1/orange/lib/python3.10/site-packages (from django-crispy-forms) (4.2.2)
Requirement already satisfied: asgiref<4,>=3.6.0 in /csc108/project1/orange/lib/python3.10/site-packages (from django>=3.2->django-crispy-forms) (3.7.2)
Requirement already satisfied: sqlparse>=0.3.1 in /csc108/project1/orange/lib/python3.10/site-packages (from django>=3.2->django-crispy-forms) (0.4.4)
Requirement already satisfied: typing-extensions>=4 in /csc108/project1/orange/lib/python3.10/site-packages (from django>=3.2->django-crispy-forms)
Installing collected packages: django-crispy-forms
Successfully installed django-crispy-forms-2.0
(orange) occc@occc-VirtualBox:/csc108/project1/mywebsite$ pip install crispy-bootstrap5
Collecting crispy-bootstrap5
  Downloading crispy_bootstrap5-0.7-py3-none-any.whl (22 kB)
Requirement already satisfied: django-crispy-forms>=1.13.0 in /csc108/project1/orange/lib/python3.10/site-packages (from crispy-bootstrap5) (2.0)
Requirement already satisfied: django>=3.2 in /csc108/project1/orange/lib/python3.10/site-packages (from crispy-bootstrap5) (4.2.2)
Requirement already satisfied: asgiref<4,>=3.6.0 in /csc108/project1/orange/lib/python3.10/site-packages (from django>=3.2->crispy-bootstrap5) (3.7.2)
Requirement already satisfied: sqlparse>=0.3.1 in /csc108/project1/orange/lib/python3.10/site-packages (from django>=3.2->crispy-bootstrap5) (0.4.4)
Requirement already satisfied: typing-extensions>=4 in /csc108/project1/orange/lib/python3.10/site-packages (from asgiref<4,>=3.6.0->django>=3.2->crispy-bootstrap5)
Installing collected packages: crispy-bootstrap5
Successfully installed crispy-bootstrap5-0.7
(orange) occc@occc-VirtualBox:/csc108/project1/mywebsite$
```

add crispy forms to our apps

add crispy forms to settings.py

```
1 INSTALLED_APPS = [
2     'django.contrib.admin',
3     'django.contrib.auth',
4     'django.contrib.contenttypes',
5     'django.contrib.sessions',
6     'django.contrib.messages',
7     'django.contrib.staticfiles',
8     'csc108',
9     'users',
10    'faculty',
11    'registrar',
12    'students',
13    "crispy_forms",
14    "crispy_bootstrap5",
15 ]
```

add crispy form attr to settings file

add crispy forms attr to settings.py

```
1 DEFAULT_AUTO_FIELD = 'django.db.models.BigAutoField'
2 AUTH_USER_MODEL = 'users.CustomUser'
3 X_FRAME_OPTIONS = "SAMEORIGIN"
4
5
6 CRISPY_ALLOWED_TEMPLATE_PACKS = "bootstrap5"
7 CRISPY_TEMPLATE_PACK = "bootstrap5"
```

edit our adduser.html template

add crispy to our template adduser.html

```

1  {% extends "registrar/base2.html" %} 
2  {% load static %} 
3  {% load crispy_forms_filters %} 
4  {% load crispy_forms_tags %} 
5 
6  {% block content %} 
7  <form action = "" method = "post"> 
8      {% csrf_token %} 
9      {{ form | crispy }} 
10     <input type="submit" value="Submit"> 
11 </form> 
12 
13 {% endblock content %}
```

{% load crispy_forms_filters %}
 {% load crispy_forms_tags %}
 and change
 {{ form | crispy }}

template filters

In Django, you can use filters within your templates to alter the value of a variable. Filters are used by placing a pipe character (|) followed by the filter name after the variable. The **form | crispy** the form is piped into the crispy filter which transforms it to have in our case bootstrap formatting.

<https://docs.djangoproject.com/en/4.2/ref/templates/builtins/#filter>

Examples:

{{ name|lower }}

In this example, the lower filter is applied to the variable **name**. This will convert the value of **name** to lowercase.

{{ my_list|length }}

length: Returns the length of the value.

```
{{ my_date|date:"D d M Y" }}
```

date: Formats a date according to the given format.

```
{{ my_var|default:"Default Value" }}
```

default: If a variable is false or empty, use given default.

```
{{ my_html_string|escape }}
```

escape: Escapes a string's HTML.

```
{{ my_string|escape|lower }}
```

In this example, the escape filter is applied first, and then the lower filter is applied to the result. Filters are always applied from left to right.

The screenshot shows a Firefox browser window with the title "College main web page". The address bar shows the URL "csc108.sunyorange.edu.local:8000/registrar/". The page content is as follows:

Registrar

You are logged in as **admin1** role: **admin**

Home

Add User

Orders

Products

Customers

Password*

Last login

Superuser status
Designates that this user has all permissions without explicitly assigning them.

Groups

The groups this user belongs to. A user will get all permissions granted to each of their groups.

User permissions

admin | log entry | Can add log entry
admin | log entry | Can change log entry
admin | log entry | Can delete log entry
admin | log entry | Can view log entry

Specific permissions for this user.

Username*
Required. 150 characters or fewer. Letters, digits and @/./+/-/_ only.

First name

Log out

The above gets us closer to creating a nice form for user add, however we have duplicates and other fields which are not necessary.

Some items that we don't need are like the superuser, lastlogin, etc, etc

If we do a describe on the users_customuser table these are the only fields and most will be populated by django automatically.

```
(orange) occc@occc-VirtualBox:/csc108/project1/mywebsite$ python manage.py dbshell
psql (14.8 (Ubuntu 14.8-0ubuntu0.22.04.1))
SSL connection (protocol: TLSv1.3, cipher: TLS_AES_256_GCM_SHA384, bits: 256, compression: off)
Type "help" for help.

orange=> \d users_customuser
          Table "public.users_customuser"
   Column    |      Type      | Collation | Nullable | Default
-----+-----+-----+-----+-----+
  id | bigint |           | not null | generated by default as identity
password | character varying(128) |           | not null |
last_login | timestamp with time zone |           |           |
is_superuser | boolean |           | not null |
username | character varying(150) |           | not null |
first_name | character varying(150) |           | not null |
last_name | character varying(150) |           | not null |
is_staff | boolean |           | not null |
is_active | boolean |           | not null |
date_joined | timestamp with time zone |           | not null |
email | character varying(254) |           | not null |
status | character varying(100) |           | not null |
description | text |           | not null |

Indexes:
    "users_customuser_pkey" PRIMARY KEY, btree (id)
    "users_customuser_email_6445acef_like" btree (email varchar_pattern_ops)
    "users_customuser_email_key" UNIQUE CONSTRAINT, btree (email)
    "users_customuser_username_80452fdf_like" btree (username varchar_pattern_ops)
    "users_customuser_username_key" UNIQUE CONSTRAINT, btree (username)

Referenced by:
    TABLE "django_admin_log" CONSTRAINT "django_admin_log_user_id_c564eba6_fk_users_customuser_id" FK
    TABLE "users_customuser_groups" CONSTRAINT "users_customuser_gro_customuser_id_958147bf_fk_users_
    TABLE "users_customuser_user_permissions" CONSTRAINT "users_customuser_use_customuser_id_5771478b_fk_u

orange=>
```

id

id is auto generated as a sequence and primary key for our table. We can for now think of it as student A-number.

password

Password*

Password*

- Your password can't be too similar to your other personal information.
- Your password must contain at least 8 characters.
- Your password can't be a commonly used password.
- Your password can't be entirely numeric.

Password confirmation*

Enter the same password as before, for verification.

Submit

From the above we see that the password is there multiple times. The last two in the form are password1 and password2. Most likely we might not want to add the passwords here, but have them generated algorithmically by django.

last login, superuser,groups

Last login

Superuser status
Designates that this user has all permissions without explicitly assigning them.

Groups

The groups this user belongs to. A user will get all permissions granted to each of their groups.

User permissions

admin | log entry | Can add log entry
admin | log entry | Can change log entry
admin | log entry | Can delete log entry
admin | log entry | Can view log entry

Specific permissions for this user.

Right now we can keep them empty and using default values. Not needed to display or change.

user name, first name, last name, staff,active,date joined,email,status,description

Username*

Required. 150 characters or fewer. Letters, digits and @/./+/-/_ only.

First name

Last name

Staff status
Designates whether the user can log into this admin site.

Active
Designates whether this user should be treated as active. Unselect this instead of deleting accounts.

Date joined*

Email*

Status*

student
student
faculty
admin

username is a must, as well as full name, Email in our model is set to not NULL so it must be filled out. Description hidden by the status drop down might be nice.

12.5.1 get all the fields in our form

From the above we have a lot of fields and dont know what they are called. We can create a small python script to display them. All the python code can be runned as standlone without django. We can even use scripts to populate the database without the need of the web browser.

12.5.2 create a standalone script to run parts of django

django stand alone script helper_scripts.py

```

1 import os
2 import django
3
4 os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'mywebsite.settings')
5 django.setup()
6
7
8
9 from django import forms
10 from django.contrib.auth.forms import UserCreationForm
11 from django.contrib.auth import get_user_model
12 from users.models import CustomUser
13
14 class AddUserForm(UserCreationForm):
15
16     class Meta:
17         model = CustomUser
18         fields = '__all__'
19
20
21 if __name__ == '__main__':
22     form = AddUserForm()
23     fields = list(form.base_fields)
24
25     for field in list(form.declared_fields):
26         if field not in fields:
27             fields.append(field)
28     print(fields)

```

```

(orange) occc@occc-VirtualBox:/csc108/project1/mywebsite$ pwd
/csc108/project1/mywebsite
(orange) occc@occc-VirtualBox:/csc108/project1/mywebsite$ ls
admin  db.sqlite3  helper_scripts.py  mywebsite  registrar  users
csc108  faculty   manage.py        occc.txt   students
(orange) occc@occc-VirtualBox:/csc108/project1/mywebsite$ python helper_scripts.py
['password', 'last_login', 'is_superuser', 'groups', 'user_permissions', 'username', 'first_name', 'last_name', 'is_staff', 'is_active', 'date_joined', 'email', 'status', 'description', 'password1', 'password2']
(orange) occc@occc-VirtualBox:/csc108/project1/mywebsite$ █

```

Our helper script has to be located in the same directory as the manage.py script. The os environment was taken as an example from the wsgi.py script in the project directory(same directory as the settings file)

```
(orange) occc@occc-VirtualBox:/csc108/project1/mywebsite$ pwd
/csc108/project1/mywebsite/mywebsite
(orange) occc@occc-VirtualBox:/csc108/project1/mywebsite/mywebsite$ ls
asgi.py __init__.py __pycache__ settings.py urls.py wsgi.py
(orange) occc@occc-VirtualBox:/csc108/project1/mywebsite/mywebsite$ more wsgi.py
"""

WSGI config for mywebsite project.

It exposes the WSGI callable as a module-level variable named ``application``.

For more information on this file, see
https://docs.djangoproject.com/en/4.2/howto/deployment/wsgi/
"""

import os

from django.core.wsgi import get_wsgi_application

os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'mywebsite.settings')

application = get_wsgi_application()
(orange) occc@occc-VirtualBox:/csc108/project1/mywebsite/mywebsite$
```

output from helper script

```
['password',
'last_login',
'is_superuser',
'groups',
'user_permissions',
'username',
'first_name',
'last_name',
'is_staff',
'is_active',
'date_joined',
'email',
'status',
'description',
'password1',
'password2']
```

12.5.3

better,faster and improved adduser-form

new registration adduser form

```
1 from django import forms
2 from django.contrib.auth.forms import UserCreationForm
3 from django.contrib.auth import get_user_model
4 from users.models import CustomUser
5
6 class AddUserForm(UserCreationForm):
7
8     class Meta:
9         model = CustomUser
10        fields = ["username", "first_name", "last_name", "email", "status", "description"]
11        #exclude = ["password1", "password2"]
12    def __init__(self, *args, **kwargs):
13        super(AddUserForm, self).__init__(*args, **kwargs)
14        #remove what you like
15        self.fields.pop('password1')
16        self.fields.pop('password2')
```

In the class Meta we limited the fields and their order by using the fields tag. However the password1 and password2 are hard coded in the **UserCreationForm** parent class and must be manually removed from this form.

You are logged in as **admin1** role: **admin**

Username*

Required. 150 characters or fewer. Letters, digits and @/./+/-/_ only.

First name

Last name

Email*

Status*

student

Description

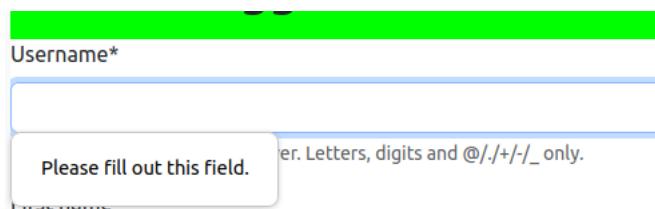
Log out

Submit

12.5.4

error checking and filling in other information

If we press submit it will give us a little pop telling us that some fields need to be filled in.



The screenshot shows a web browser window with the title "College main web page". The URL is "csc108.sunyorange.edu.local:8000/registrar/". The page is titled "Registrar" and features a sidebar with links: Home (highlighted in blue), Add User, Orders, Products, and Customers.

The main content area displays a success message: "You are logged in as **admin1** role: admin". Below this, there is a form for adding a new user:

- Username***: mkrrajca (with a note: Required. 150 characters or fewer. Letters, digits and @./+/-/_ only.)
- First name**: Miroslav
- Last name**: Krajca
- Email***: mk@someplace.com
- Status***: student
- Description**: stuff

At the bottom left is a "Log out" link, and at the bottom right is a "Submit" button.

You are logged in as **admin1** role: **admin**

KeyError at /registrar/adduser/

'password1'

Request Method: POST
Request URL: http://csc108.sunyorange.edu.local:8000/registrar/adduser/
Django Version: 4.2.2
Exception Type: KeyError
Exception Value: 'password1'
Exception Location: /csc108/project1/orange/lib/python3.10/site-packages/django/contrib/auth/forms.py, line 141, in save
Raised during: registrar.views.adduser
Python Executable: /csc108/project1/orange/bin/python
Python Version: 3.10.6
Python Path: ['/csc108/project1/mywebsite', '/usr/lib/python310.zip', '/usr/lib/python3.10', '/usr/lib/python3.10/lib-dynload', '/csc108/project1/orange/lib/python3.10/site-packages']
Server time: Fri, 07 Jul 2023 13:14:35 +0000

Traceback [Switch to copy-and-paste view](#)

```
/csc108/project1/orange/lib/python3.10/site-packages/django/core/handlers/exception.py, line 55, in inner
  55.         response = get_response(request)
▶ Local vars

/csc108/project1/orange/lib/python3.10/site-packages/django/core/handlers/base.py, line 197, in _get_response
  197.         response = wrapped_callback(request, *callback_args, **callback_kwargs)
▶ Local vars

/csc108/project1/orange/lib/python3.10/site-packages/django/contrib/auth/decorators.py, line 23, in _wrapper_view
  23.         return view_func(request, *args, **kwargs)
▶ Local vars

/csc108/project1/mywebsite/registrar/views.py, line 29, in adduser
  29.         form.save()
▶ Local vars

/csc108/project1/orange/lib/python3.10/site-packages/django/contrib/auth/forms.py, line 141, in save
  141.             raise ValueError('The two password fields didn't match')


```

Log out

Some fields have to be filled in order for the form to be validated and eventually saved to our database.

We can manually add them in after the submit was done by the user.

<https://docs.djangoproject.com/en/4.2/topics/forms/modelforms/#the-self-cleaning-forms>

12.5.5 Django User Authentication in form

By default the build in authentication system requires at least two fields to be present in order to generate a password. These fields are password1 and password2

which it checks that they are the same and generates a password based on the information present. We want to auto generate a password in our view, however the form.is_valid() is expecting these fields to be there. We can add them as hidden fields and pre-populate them with garbage information. The password has to be at least 8 characters long.

hiddent fileds in form

```

1 from django import forms
2 from django.contrib.auth.forms import UserCreationForm
3 from django.contrib.auth import get_user_model
4 from users.models import CustomUser
5
6 class AddUserForm(UserCreationForm):
7     password1 = forms.CharField(widget=forms.HiddenInput(), initial="↖
8         ↴ !&!!!!123!!!!occc!!!12345")
9     password2 = forms.CharField(widget=forms.HiddenInput(), initial="↖
10        ↴ !&!!!!123!!!!occc!!!12345")
11     class Meta:
12         model = CustomUser
13         fields = ["username", "first_name", "last_name", "email", "status", "↖
14             ↴ description"]

```

modify registrar view.py

hiddent fileds in form

```

1 from django.shortcuts import render, redirect
2 from django.http import HttpResponseRedirect
3 from django.template import loader
4 from django.contrib.auth.decorators import login_required
5
6 from .forms import AddUserForm
7 from django.contrib import messages
8
9 def index(request):
10     """ main landing page for or website """
11     return render(request, "registrar/home.html")
12 @login_required
13 def home(request):
14     if(request.session['role'] != 'admin'):
15         return HttpResponseRedirect('401 Unauthorized', status=401)
16         return render(request, '404.html', status=404)
17     else:
18         return render(request, "registrar/home.html")
19
20 @login_required
21 def adduser(request):
22     flag = 0
23     if(request.session['role'] != 'admin'):
24         return HttpResponseRedirect('401 Unauthorized', status=401)
25     else:

```

```
1
26     if request.POST: #submitting a completed form
27         flag = 1
28         form = AddUserForm(request.POST)
29
30         if form.is_valid():
31
32             temp_form = form.save(commit=False)
33             temp_form.set_password("temp123!123!")
34
35             temp_form.save()
36             #since we are in an i frame
37             return redirect('emptypage')
38
39         else:
40             for error in list(form.errors.values()):
41                 messages.error(request, error)
42
43             #not a post but a get. getting the form for first time
44             #or something failed in the is_valid() or save
45             context = {}
46             if(flag == 1):
47                 context['form'] = form
48             else:
49                 context['form'] = AddUserForm()
50
51             return render(request,"registrar/adduser.html",context)
52
53 @login_required
54 def emptypage(request):
55     return render(request,"registrar/emptypage.html")
```

```
temp_form = form.save(commit=False)
temp_form.set_password("temp123!123!")
temp_form.save()
```

This will allow us to add a password that was not entered on the form page. In real situation this will be some password that was generated as initial password that the user would then change on first login in and be something that the user would know like their username plus birthday or something similar. The save function actually writes the information to the database.

Change user

mkrajca

Username: Required. 150 characters or fewer. Letters, digits and @/./+/-/_ only.

Password:
algorithm: pbkdf2_sha256 **iterations:** 60000 **salt:** z3Rsn8***** **hash:** 477hFv*****
Raw passwords are not stored, so there is no way to see this user's password, but you can change the password using this form.

Personal info

First name:

Last name:

Email:

Permissions

Active
Designates whether this user should be treated as active. Unselect this instead of deleting accounts.

Staff status
Designates whether the user can log into this admin site.

Superuser status
Designates that this user has all permissions without explicitly assigning them.

Groups:

Available groups	Chosen groups
<input type="text" value="Filter"/>	<input type="text" value="Filter"/>
	<input style="float: right;" type="button" value="+"/>

Choose all **Remove all**
The groups this user belongs to. A user will get all permissions granted to each of their groups. Hold down "Control", or "Command" on a Mac, to select more than one.

User permissions:

12.6 ➤ using django debug toolbar

We can install a debug tool bar to see the sql and other pertinent information about our website. However it will not work with the nested site we have not. the iframe will duplicate the tool bar and prevent up from using it. If we want to use it we have to remove the "iframe".

12.6.1

remove the navbar javascript from base.html

new base.html

```
1  {% load static %} 
2  <!DOCTYPE html>
3  <html lang="en-US">
4      <head>
5          <meta charset="utf-8" >
6          <title>College main web page</title>
7          <link rel="stylesheet" type="text/css" href="{% static 'csc108/bootstrap-5.3.0-dist/css/bootstrap.min.css' %}">
8          <link rel="stylesheet" type="text/css" href="{% static 'registrar/navbar.css' %}">
9
10
11     </head>
12     <body>
13         <main>
14             {% include 'registrar/includes/navbar.html' %}
15             <div style="width: 100%;">
16                 {% include 'csc108/includes/messaging.html' %}
17                 {% block content %}
18                 {% endblock %}
19
20             </div>
21         </main>
22
23     <!-- Optional Javascript -->
24     <script src="{% static 'csc108/jquery3.7.0/jquery-3.7.0.min.js' %}">
25         </script>
26     <script src="{% static 'csc108/bootstrap-5.3.0-dist/js/bootstrap.bundle.min.js' %}"></script>
27     <script src="{% static 'registrar/navbar.js' %}"></script>
28
29     <script>
30         $(document).ready(function(){
31             window.setTimeout(function() {
32                 $(".alert").fadeTo(500, 0).slideUp(500, function(){
33                     $(this).remove();
34                 });
35             }, 5000);
36     </script>
```

```
35 });
36
37
38 </script>
39
40
41 </body>
42 </html>
```

12.6.2

redesign the registrar home page navbar

new navbar

```
1 <div class="d-flex flex-column flex-shrink-0 p-3 bg-light" style="width: ↴
2   ↴ 280px;">
3   <a href="" class="d-flex align-items-center mb-3 mb-md-0 me-md-auto ↴
4     ↴ link-dark text-decoration-none">
5     
7     <span class="fs-4">Registrar</span>
8   </a>
9   <hr>
10  <ul class="nav nav-pills flex-column mb-auto">
11    <li class="nav-item">
12      <a href="/registrar" class="nav-link active" aria-current="page">
13        <svg class="bi me-2" width="16" height="16"><use xlink:href="# ↴
14          ↴ home"/></svg>
15        Home
16      </a>
17    </li>
18    {% if inadduser == 1 %}
19    <li>
20      <svg class="bi me-2" width="16" height="16"><use xlink:href="# ↴
21          ↴ speedometer2"/></svg>
22      Add User
23    </li>
24    {% else %}
25    <a href="adduser" class="nav-link link-dark">
26      <svg class="bi me-2" width="16" height="16"><use xlink:href="# ↴
27          ↴ speedometer2"/></svg>
28      Add User
29    </a>
30    {% endif %}
31    <li>
32      <a href="#" class="nav-link link-dark">
33        <svg class="bi me-2" width="16" height="16"><use xlink:href="# ↴
34          ↴ table"/></svg>
35      Orders
36    </a>
37  </li>
```

```

31      <li>
32          <a href="#" class="nav-link link-dark">
33              <svg class="bi me-2" width="16" height="16"><use xlink:href="#grid"></svg>
34          Products
35      </a>
36  </li>
37  <li>
38      <a href="#" class="nav-link link-dark">
39          <svg class="bi me-2" width="16" height="16"><use xlink:href="#people-circle"></svg>
40      Customers
41      </a>
42  </li>
43 </ul>
44 <hr>
45 <div class="dropdown">
46
47     {% if user.is_authenticated %}
48         <a class="nav-item nav-link" href="{% url 'logout' %}">
49             Log out <i class="fas fa-sign-out-alt"></i>
50         </a>
51     {% else %}
52         <a class="nav-item nav-link" href="{% url 'login' %}">
53             Log in <i class="fas fa-sign-in-alt"></i>
54         </a>
55     {% endif %}
56
57
58
59     </div>
60 </div>

```

12.6.3

make small changes to our view to disable the adduser link when in adduser

updated adduser view

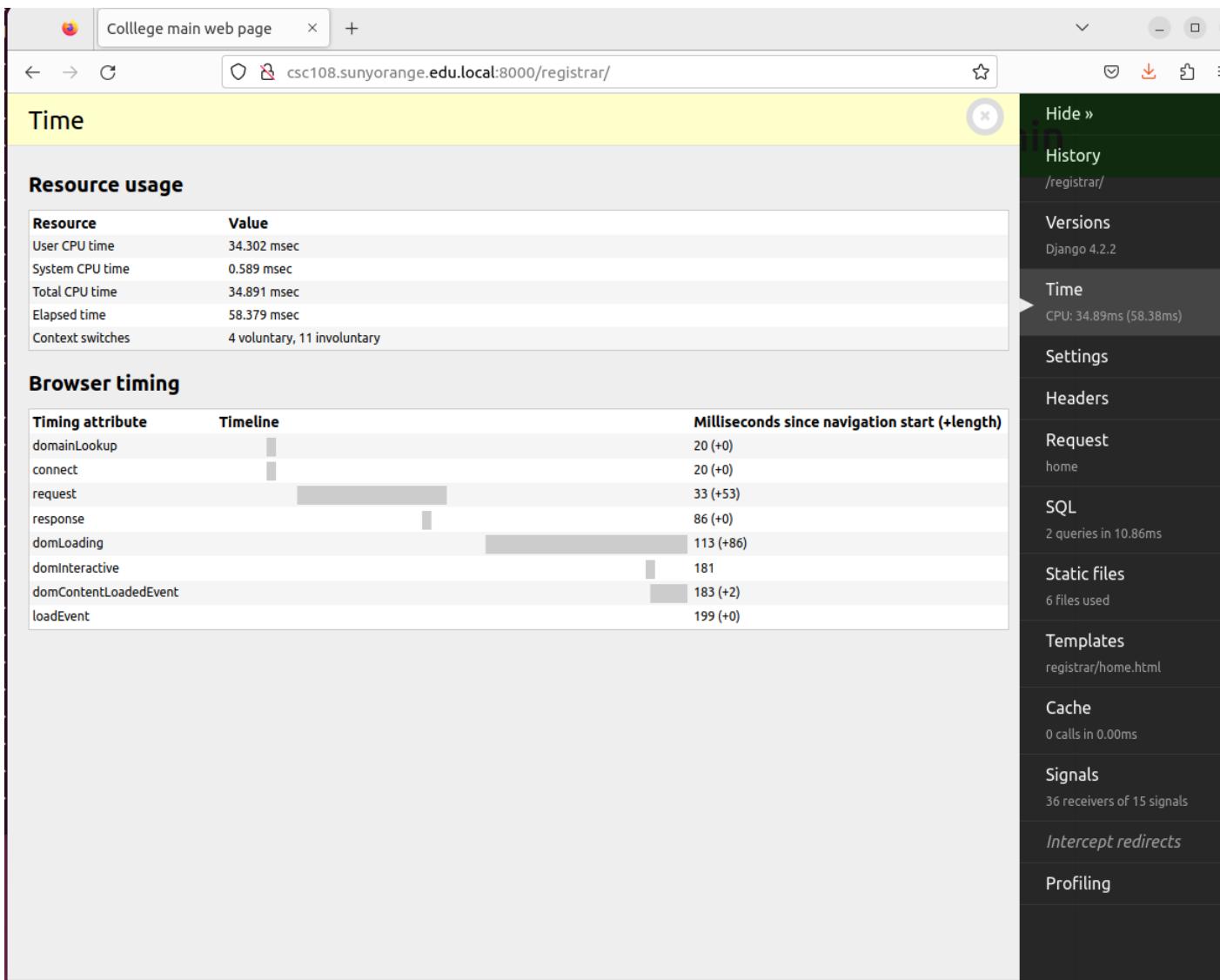
```

1 @login_required
2 def adduser(request):
3     flag = 0
4     if(request.session['role'] != 'admin'):
5         return HttpResponseRedirect('401 Unauthorized', status=401)
6     else:
7         if request.POST: #submitting a completed form
8             flag = 1
9             form = AddUserForm(request.POST)
10

```

```
11     if form.is_valid():
12
13         temp_form = form.save(commit=False)
14         temp_form.set_password("temp123!123!")
15
16         temp_form.save()
17         return redirect('registrar')
18     else:
19         for error in list(form.errors.values()):
20             messages.error(request, error)
21
22     #not a post but a get. getting the form for first time
23     #or something failed in the is_valid() or save
24     context = {}
25     context['inadduser'] = 1
26     if(flag == 1):
27         context['form'] = form
28     else:
29         context['form'] = AddUserForm()
30
31
32     return render(request,"registrar/adduser.html",context)
```

Now we can use the debug toolbar.
BEGUG TOOLBAR SHOULD ONLY BE USED IN DEVELOPMENT AND
NEVR IN PRODUCTION!!!!



12.7 display all users

12.7.1 create an allusers view

updated registrar views.py

```

1 @login_required
2 def allusers(request):
3     if(request.session['role'] != 'admin'):
4         return HttpResponseRedirect('401 Unauthorized', status=401)
5     else:
6         queryset = CustomUser.objects.all()
7         context={}
8         context['query'] = queryset

```

```
9     context['inallusers'] = 1
10    return render(request, 'registrar/allusers.html', context)
```

12.7.2 allusers.html

registrar allusers.html

```
1  {% extends "registrar/base.html" %}          → registrar allusers.html
2  {% load static %}                            → registrar allusers.html
3  {% load crispy_forms_filters %}              → registrar allusers.html
4  {% load crispy_forms_tags %}                  → registrar allusers.html
5
6  {% block content %}                         → registrar allusers.html
7  <table class="table table-striped">           → registrar allusers.html
8      <thead>                                → registrar allusers.html
9          <tr>                                 → registrar allusers.html
10             <th scope="col">id</th>            → registrar allusers.html
11             <th scope="col">username</th>        → registrar allusers.html
12             <th scope="col">First</th>           → registrar allusers.html
13             <th scope="col">Last</th>            → registrar allusers.html
14             <th scope="col">email</th>           → registrar allusers.html
15         </tr>                                 → registrar allusers.html
16     </thead>                                → registrar allusers.html
17     <tbody>                                 → registrar allusers.html
18         {% for q in query %}                 → registrar allusers.html
19             <tr>                               → registrar allusers.html
20                 <th scope="row">{{ q.id }}</th>   → registrar allusers.html
21                 <td>{{ q.username }}</td>       → registrar allusers.html
22                 <td>{{ q.first_name }}</td>     → registrar allusers.html
23                 <td>{{ q.last_name }}</td>       → registrar allusers.html
24                 <td>{{ q.email }}</td>          → registrar allusers.html
25         </tr>                                → registrar allusers.html
26
27     {% endfor %}                           → registrar allusers.html
28
29     </tbody>                                → registrar allusers.html
30 </table>                                 → registrar allusers.html
31
32 {% endblock content %}
```

12.7.3 all allusers to navbar

updated registrar navbar

```

1   {% if inallusers == 1 %}
2     <li>
3       <svg class="bi me-2" width="16" height="16"><use xlink:href="#table"/></svg>
4         allusers
5     </li>
6     {% else %}
7     <li>
8       <a href="allusers" class="nav-link link-dark">
9         <svg class="bi me-2" width="16" height="16"><use xlink:href="#table"/></svg>
10        allusers
11      </a>
12    </li>
13  {% endif %}
```

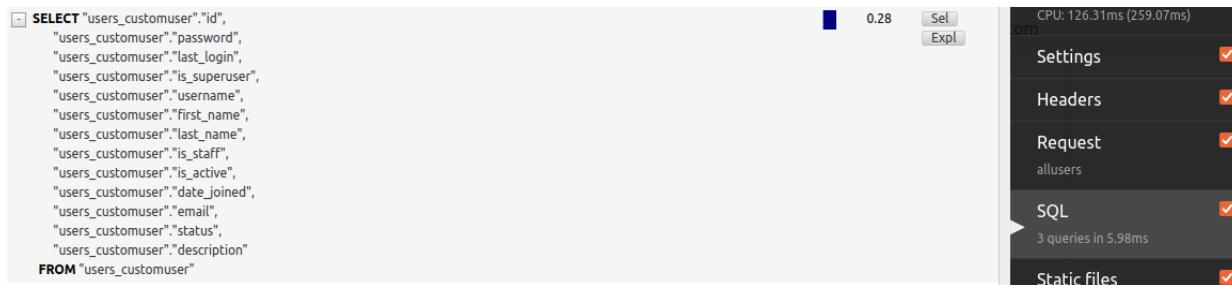
12.7.4 add urls to urls.py

updated registrar urls.py

```

1 from django.urls import path
2
3 from . import views
4
5 urlpatterns = [
6   path("registrar/", views.home, name="registrar"),
7   path("registrar/adduser/", views.adduser, name="adduser"),
8   path("registrar/allusers/",views.allusers,name="allusers"),
9   path("emptypage",views.emptypage,name="emptypage")
10 ]
```

	id	username	First	Last	email
10	teacher1	staff1	staff1		teacher1@localhost
5	student1	jane	Doe		student1@localhost
1	occc				occc@localhost
9	admin1	admin	admin		admin1@localhost
16	mkrajca	Miroslav	Krajca		mk@someplace.com



```

[{"id": "1", "label": "0.28", "x": 650, "y": 50}, {"id": "2", "label": "Sel", "x": 700, "y": 50}, {"id": "3", "label": "Expl", "x": 750, "y": 50}, {"id": "4", "label": "CPU: 126.31ms (259.07ms)", "x": 800, "y": 60}, {"id": "5", "label": "Settings", "x": 850, "y": 70}, {"id": "6", "label": "Headers", "x": 850, "y": 85}, {"id": "7", "label": "Request", "x": 850, "y": 100}, {"id": "8", "label": "allusers", "x": 850, "y": 115}, {"id": "9", "label": "SQL", "x": 850, "y": 130}, {"id": "10", "label": "3 queries in 5.98ms", "x": 850, "y": 145}, {"id": "11", "label": "Static files", "x": 850, "y": 160}

```

`SELECT "users_customuser"."id",
"users_customuser"."password",
"users_customuser"."last_login",
"users_customuser"."is_superuser",
"users_customuser"."username",
"users_customuser"."first_name",
"users_customuser"."last_name",
"users_customuser"."is_staff",
"users_customuser"."is_active",
"users_customuser"."date_joined",
"users_customuser"."email",
"users_customuser"."status",
"users_customuser"."description"
FROM "users_customuser"`

12.8 ORM queryset

```
queryset = CustomUser.objects.all()
```

Model Managers

A model manager is the interface for retrieving objects from the database via a QuerySet. Every model has, by default, at least one Manager with the name objects.

Our model is CustomUser which has a model manager that is responsible for saving, updating and retrieving the data from the database table that is storing data for our model.

Object is nothing more than an instance of a class. In our case the objects will represent a list of class instances and each instance will represent a row in our table.

The .all() is a filter to specify that we want all the rows in the table to be stored in a list and each item in the list will be a class object representing one row of the table.

helper-script.py

```

1 import os
2 import django
3
4 os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'mywebsite.settings')
5 django.setup()
6
7
8
9
10 from django import forms
11 from django.contrib.auth.forms import UserCreationForm
12 from django.contrib.auth import get_user_model
13 from users.models import CustomUser

```

```
14
15 class AddUserForm(UserCreationForm):
16
17     class Meta:
18         model = CustomUser
19         fields = '__all__'
20
21
22 if __name__ == '__main__':
23     queryset = CustomUser.objects
24     print("-" * 30)
25     print("Object no filter type")
26     print("-" * 30)
27     print(type(queryset))
28     print("-" * 30)
29     print("Object no filter palain queryset")
30     print("-" * 30)
31     print(queryset)
32     queryset = CustomUser.objects.all()
33     print("-" * 30)
34     print("plain queryset type using all filter")
35     print("-" * 30)
36     print(type(queryset))
37     print("-" * 30)
38     print("plain queryset using all filter")
39     print("-" * 30)
40     print(queryset)
41     print("-" * 30)
42     print("plain queryset first resultset item")
43     print("-" * 30)
44     print(queryset[0])
45     print("-" * 30)
46     print("plain queryset first resultset item type")
47     print("-" * 30)
48     print(type(queryset[0]))
49     print("-" * 30)
50     print("plain queryset first resultset class object id")
51     print("-" * 30)
52     print(queryset[0].id)
53     print("-" * 30)
54     print("plain queryset first resultset class all fields")
55     print("-" * 30)
56     print(queryset[0]._meta.fields)
```

```
(orange) occc@occc-VirtualBox:/csc108/project1/mywebsite$ python helper_scripts.py
-----
Object no filter type
-----
<class 'django.contrib.auth.models.UserManager'>
-----
Object no filter plain queryset
-----
users.CustomUser.objects
-----
plain queryset type using all filter
-----
<class 'django.db.models.query.QuerySet'>
-----
plain queryset using all filter
-----
<QuerySet [<CustomUser: teacher1>, <CustomUser: student1>, <CustomUser: occc>,
<CustomUser: admin1>, <CustomUser: mkrajca>]>
-----
plain queryset first resultset item
-----
teacher1
-----
plain queryset first resultset item type
-----
<class 'users.models.CustomUser'>
-----
plain queryset first resultset class object id
-----
10
-----
plain queryset first resultset class all fields
-----
(<django.db.models.fields.BigAutoField: id>,
<django.db.models.fields.CharField: password>,
<django.db.models.fields.DateTimeField: last_login>,
<django.db.models.fields.BooleanField: is_superuser>,
<django.db.models.fields.CharField: username>,
<django.db.models.fields.CharField: first_name>,
<django.db.models.fields.CharField: last_name>,
<django.db.models.fields.BooleanField: is_staff>,
<django.db.models.fields.BooleanField: is_active>,
<django.db.models.fields.DateTimeField: date_joined>,
<django.db.models.fields.EmailField: email>,
<django.db.models.fields.CharField: status>,
<django.db.models.fields.TextField: description>)

(orange) occc@occc-VirtualBox:/csc108/project1/mywebsite$
```

why did the users names came up by default

```
<QuerySet [<CustomUser: teacher1>,
<CustomUser: student1>,
<CustomUser: occc>,
```

```
<CustomUser: admin1>,
<CustomUser: mkrajca]>
```

user model

```
1 from django.contrib.auth.models import AbstractUser
2 from django.db import models
3 import uuid
4
5 class CustomUser(AbstractUser):
6
7     STATUS = (
8         ('student', 'student'),
9         ('faculty', 'faculty'),
10        ('admin', 'admin'),
11    )
12
13    email = models.EmailField(unique=True)
14    status = models.CharField(max_length=100, choices=STATUS, default='regular')
15    description = models.TextField("Description", max_length=600, default='', blank=True)
16
17    def __str__(self):
18        return self.username
```

```
def __str__(self):
    return self.username
```

By default it will display the user name.

12.8.1 Allusers Java script and JSON data

Create new web page that will inherit base.html

alluserjs.html

```
1 {% extends "registrar/base.html" %}
2 {% load static %}
3 {% load crispy_forms_filters %}
4 {% load crispy_forms_tags %}
5 {% block title %}
6     All users Page Javascript using jsonp
7 {% endblock title %}
8 {% block custom_css %}
9     <link rel="stylesheet"
10        type="text/css"
11        href="{% static 'mainpage/jquery-ui-1.13.2/jquery-ui.css' %}" />
12     <link rel="stylesheet"
13        href="{% static 'mainpage/grid-3.5.1/pqgrid.min.css' %}" />
```

```
14    <link rel="stylesheet"
15        href="{% static 'mainpage/grid-3.5.1/themes/Office/pqgrid.css' %}" ↵
16        />
17    {% endblock custom_css %}
18    {% block content %}
19        <div id="grid_jsonp" style="margin:5px auto;"></div>
20    {% endblock content %}
21    {% block custom_js %}
22        <script src="http://ajax.googleapis.com/ajax/libs/jquery/1.8.3/jquery.min.js"></script>
23        <script src="{% static 'mainpage/jquery-ui-1.13.2/jquery-ui.min.js' %}"></script>
24        <script src="{% static 'mainpage/grid-3.5.1/pqgrid.min.js' %}"></script>
25        <script>
26            $(function() {
27                var colModel = [
28                    {
29                        title: "ID",
30                        dataType: "integer",
31                        dataIndx: "id",
32                        editable: false,
33                        width: 80
34                    },
35                    {
36                        title: "User Name",
37                        width: 165,
38                        dataType: "string",
39                        dataIndx: "username"
40                    },
41                    {
42                        title: "First name",
43                        width: 140,
44                        dataType: "string",
45                        align: "right",
46                        dataIndx: "first_name"
47                    },
48                    {
49                        title: "Last Name",
50                        width: 100,
51                        dataType: "string",
52                        align: "right",
53                        dataIndx: "last_name"
54                    },
55                    {
56                        title: "Email",
57                        width: 100,
58                        dataType: "string",
59                        align: "right",
60                        dataIndx: "email"
61                ];
62                var dataModel = {
63                    location: "remote",
64                    dataType: "json",
65                    method: "GET",
66                    url: "/registrar/allusersjsonp/"
67                };
68
69                $("#grid_jsonp").pqGrid({
70                    height: 450,
71                    scrollModel: {
72                        autoFit: true
73                });
74            }
75        });
76    
```

```

68 },
69     dataModel: dataModel,
70     colModel: colModel,
71     numberCell: {
72         resizable: true,
73         width: 30,
74         title: "#"
75     },
76     title: "Web Development 1 Users",
77     resizable: true
78 );
79 });
80 </script>
81 {% endblock custom_js %}

```

12.8.2 modify views.py for JSON views

modify views.py

```

1 @login_required
2 def allusersjs(request):
3     if(request.session['role'] != 'admin'):
4         return HttpResponseRedirect('401 Unauthorized', status=401)
5     else:
6         return render(request, 'registrar/allusersjs.html')
7
8 @login_required
9 def allusersjsonp(request):
10    if(request.session['role'] != 'admin'):
11        return HttpResponseRedirect('401 Unauthorized', status=401)
12    else:
13        data = list(CustomUser.objects.values('id','username','first_name',
14                                         'last_name','email'))
15        ret_data ={"data":data}
16        return JsonResponse(ret_data,safe = False)

```

```
list(CustomUser.objects.values('id','username','first_name','last_name','email'))
```

Above select all records and stores them in a dictionary(values part) the parameters to the values function the columns we want to limit it to. Be default it does "SELECT *"

Then we turn those dictionaries into a list and append "data" as the main key.(required by dqgrid)

In Django, the JsonResponse class is a subclass of HttpResponseRedirect that helps to create a JSON-encoded response. It's most commonly used in AJAX requests

or whenever you need to return a JSON response from your Django view to the client side.

JsonResponse takes a Python data structure (like a list or a dictionary) and converts it into a JSON string that can be returned as an HTTP response. This makes it very useful when dealing with API views or any kind of data processing where the client-side expects JSON data.

Example:

JsonResponse example

```
1 from django.http import JsonResponse
2
3 def my_view(request):
4     data = {
5         'name': 'John',
6         'age': 30,
7         'city': 'Middletown'
8     }
9     return JsonResponse(data)
```

In this case, the data dictionary is converted into a JSON response. The client would receive the following JSON data:

JSON response client view

```
{
    "name": "John",
    "age": 30,
    "city": "Middletown"
}
```

12.8.3 new URLs

registrar urls.py

```
1 from django.urls import path
2
3 from . import views
4
5 urlpatterns = [
6     path("registrar/", views.home, name="registrar"),
```

```

7     path("registrar/adduser/", views.adduser, name="adduser"),
8     path("registrar/allusers/",views.allusers,name="allusers"),
9     path("registrar/allusersjs/",views.allusersjs,name="allusers"),
10    path("registrar/allusersjsonp/",views.allusersjsonp,name="alluserjsonp"),
11    ↴ ""),
12    path("emptypage",views.emptypage,name="emptypage"),
13 ]

```

The screenshot shows a web application interface. On the left is a sidebar with a logo, navigation links (Home, Add User, allusers, alluserjson, Customers), and a Log out button. The main content area has a title 'Web Development 1 Users' and a table with the following data:

#	ID User Name	First name	Last Name	Email
1	1 occc			occc@localhost
2	10 teacher1	staff1	teacher1	teacher1@localhost
3	9 admin1	admin	admin	admin1@localhost
4	5 student1	jane	Doe	student1@localhost
5	16 mkrajca	Miroslav	Krajca	mk@someplace.com

12.8.4 modify query set to sort items

```

data = list(CustomUser.objects.order_by('id')
|   |   |
|   .values('id','username','first_name','last_name','email'))
ret_data ={"data":data}

```

The screenshot shows a web application interface. On the left is a sidebar with a logo, navigation links (Home, Add User, allusers, alluserjson, Customers), and a Log out button. The main content area has a title 'Web Development 1 Users' and a table with the following data:

#	ID User Name	First name	Last Name	Email
1	1 occc			occc@localhost
2	5 student1	jane	Doe	student1@localhost
3	9 admin1	admin	admin	admin1@localhost
4	10 teacher1	staff1	staff1	teacher1@localhost
5	16 mkrajca	Miroslav	Krajca	mk@someplace.com

12.9 Create the rest of our models

Depending on the design the models should probably be stored in one app and not spread out to many apps.



The easiest to implement is the classroom. It does not have any dependencies on other tables. The order in which the models appear in the `models.py` does matter. Django will not be able to create a table that has foreign key dependency on a

table that does not exist.

classroom model

```
1 from django.contrib.auth.models import AbstractUser
2 from django.db import models
3
4 from django.core.exceptions import ValidationError
5
6 def validate_room_size(value):
7     if (value < 200 and value > 0):
8         return value
9     else:
10         raise ValidationError("Room size is not valid")
11
12
13
14
15 class CustomUser(AbstractUser):
16
17     STATUS = (
18         ('student', 'student'),
19         ('faculty', 'faculty'),
20         ('admin', 'admin'),
21     )
22
23     email = models.EmailField(unique=True)
24     status = models.CharField(max_length=100, choices=STATUS, default='regular')
25     description = models.TextField("Description", max_length=600, default='', blank=True)
26     def __str__(self):
27         return self.username
28
29 class classroom(models.Model):
30     roomtype = models.CharField(max_length = 100)
31     capacity = models.IntegerField(validators =[validate_room_size])
32     class_room_id = models.AutoField(auto_created = True,
33                                         primary_key = True,
34                                         serialize = False,
35                                         verbose_name ='class_room_id')
36     location = models.CharField(max_length = 30)
37     room_number = models.CharField(max_length = 30)
38
39
40
41 # Create your models here.
```

validators

<https://docs.djangoproject.com/en/4.2/ref/forms/validation/>

```
python manage.py makemigrations
python manage.py migrate
```

```
(orange) occc@occc-VirtualBox:/csc108/project1/mywebsite/users$ vi models.py
(orange) occc@occc-VirtualBox:/csc108/project1/mywebsite/users$ cd ..
(orange) occc@occc-VirtualBox:/csc108/project1/mywebsite$ python manage.py makemigrations
Migrations for 'users':
  users/migrations/0002_classroom.py
    - Create model classroom
(orange) occc@occc-VirtualBox:/csc108/project1/mywebsite$ python manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, sessions, users
Running migrations:
  Applying users.0002_classroom... OK
(orange) occc@occc-VirtualBox:/csc108/project1/mywebsite$ █
```

12.9.1

optionally register the model with our admin site

classroom model register with admin site

```
1 from django.contrib import admin
2 from django.contrib.auth.admin import UserAdmin
3 from .models import CustomUser, classroom
4
5
6 from django.contrib import admin
7 from .models import CustomUser
8 from .forms import UserRegistrationForm
9 from django.contrib.auth.admin import UserAdmin
10
11 # Register your models here.
12 class CustomUserAdmin(UserAdmin):
13     model = CustomUser
14     add_form = UserRegistrationForm
15     fieldsets = (
16         *UserAdmin.fieldsets,
17         (
18             'Other Personal info',
19             {
20                 'fields': (
21                     'status',
22                 )
23             }
24         )
25     )
26
27
28 admin.site.register(CustomUser, CustomUserAdmin)
29 class classroomAdmin(admin.ModelAdmin):
30
31     model = classroom
32     list_display = [f.name for f in classroom._meta.fields]
```

```

33
34 admin.site.register(classroom, classroomAdmin)
35
36 # Register your models here.

```

Make sure that you import the model before registering it.

Site administration

The screenshot shows the Django Site Administration interface. At the top, there's a blue header bar labeled "AUTHENTICATION AND AUTHORIZATION". Below it, a "Groups" section has "Add" and "Change" buttons. The main content area is titled "USERS" with two sub-sections: "Classrooms" and "Users". Each has its own "Add" and "Change" buttons.

<https://docs.djangoproject.com/en/4.2/ref/contrib/admin/filters/>

The screenshot shows the "Add classroom" page in the Django Admin. The URL in the browser is "csc108.sunyorange.edu.local:8000/admin/users/classroom/add/". The page title is "Django administration". On the left, a sidebar lists "AUTHENTICATION AND AUTHORIZATION" and "USERS" sections, with "Classrooms" currently selected. The main form is titled "Add classroom" and contains fields for "Roomtype:" (with a red border around the input field), "Capacity:" (with a dropdown menu), "Location:" (an empty input field), and "Room number:" (an empty input field). At the bottom are buttons for "SAVE", "Save and add another", and "Save and continue editing".

12.9.2 verify table creation

```
orange=> \d users_classroom
Table "public.users_classroom"
 Column |      Type      | Collation | Nullable | Default
-----+-----+-----+-----+-----+
roomtype | character varying(100) |           | not null |
capacity | integer          |           | not null |
class_room_id | integer |           | not null | generated by default as identity
location  | character varying(30) |           | not null |
room_number | character varying(30) |           | not null |
Indexes:
"users_classroom_pkey" PRIMARY KEY, btree (class_room_id)

orange=>
```

12.9.3 modify model

Lets assume a new requirement came in that the location field is going to be the building and that we need to have a filed indicating the campus. We can add new fields and other restrictions in our model. As long as we don't change items like primary key where data already existed we can modify our model and add or delete fields as long as referential integrity is not violated.

classroom model add fields

```
1 class classroom(models.Model):
2     CAMPUS = (
3         ("Middletown", "Middletown"),
4         ("Newburgh", "Newburgh"),
5         ("Other", "Other"),
6     )
7
8     roomtype = models.CharField(max_length = 100)
9     capacity = models.IntegerField(validators =[validate_room_size])
10    class_room_id = models.AutoField(auto_created = True,
11                                      primary_key = True,
12                                      serialize = False,
13                                      verbose_name ='class_room_id')
14    location = models.CharField(max_length = 30)
15    room_number = models.CharField(max_length = 30)
16    campus = models.CharField(max_length = 30,choices=CAMPUS,default="`"
17                             ↴ Middletown")
```

```
(orange) occc@occc-VirtualBox:/csc108/project1/mywebsite$ vi users/models.py
(orange) occc@occc-VirtualBox:/csc108/project1/mywebsite$ python manage.py makemigrations
Migrations for 'users':
  users/migrations/0004_classroom_campus.py
    - Add field campus to classroom
(orange) occc@occc-VirtualBox:/csc108/project1/mywebsite$ python manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, sessions, users
Running migrations:
  Applying users.0004_classroom_campus... OK
```

```
(orange) occc@occc-VirtualBox:/csc108/project1/mywebsite$ python manage.py dbshell
psql (14.8 (Ubuntu 14.8-0ubuntu0.22.04.1))
SSL connection (protocol: TLSv1.3, cipher: TLS_AES_256_GCM_SHA384, bits: 256, compression: off)
Type "help" for help.
```

```
orange=> \d users_classroom
```

Column	Type	Collation	Nullable	Default
roomtype	character varying(100)		not null	
capacity	integer		not null	
class_room_id	integer		not null	generated by default as identity
location	character varying(30)		not null	
room_number	character varying(30)		not null	
campus	character varying(30)		not null	

Indexes:

```
"users_classroom_pkey" PRIMARY KEY, btree (class_room_id)
```

```
orange=>
```

The screenshot shows the Django admin interface for adding a new classroom. The URL in the browser is `csc108.sunyorange.edu.local:8000/admin/users/classroom/add/`. The left sidebar has a 'Classrooms' link under the 'Users' section. The main content area is titled 'Add classroom' and contains the following fields:

- Roomtype:** A text input field.
- Capacity:** A dropdown menu.
- Location:** A text input field.
- Room number:** A text input field.
- Campus:** A dropdown menu set to 'Middletown'.

At the bottom of the form are three buttons: 'SAVE', 'Save and add another', and 'Save and continue editing'.

roomtype = models.CharField(max_length = 100)

In Django, **CharField** is a field type used in models to store small to large-sized strings. When you define a CharField in your model, Django will use it to create a **VARCHAR** column in your SQL database.

In our model, roomtype is a CharField that can hold a string of up to 100 characters. The `max_length` argument is required, and it specifies the maximum length of the characters that this field can hold in the database.

CharField and all other field types in Django come with a number of built-in options like null, blank, choices, default, help_text, primary_key, unique etc, which allows you to control the behavior of the field.

In the case of CharField, you might often see it used with an additional choices parameter, which restricts the field's value to a certain set of choices. For example:

```
CAMPUS = (
    ("Middletown", "Middletown"),
    ("Newburgh", "Newburgh"),
)
```

In this case the filed can only contain the above choices.
We could have rewrote this as:

```
CAMPUS = (
    ("M", "Middletown"),
    ("N", "Newburgh"),
)
```

```
campus = models.CharField(max_length = 1,choices=CAMPUS,default="M")
```

In that case the filed could only contain "M" or "N" and would be one character wide.

Examples:

```
field_name = models.CharField(max_length=200, **options)
```

Field Options	Description
Null	If True, Django will store empty values as NULL in the database. Default is False.
Blank	If True, the field is allowed to be blank. Default is False.
db_column	The name of the database column to use for this field. If this isn't given, Django will use the field's name.
db_index	If True, a database index will be created for this field
db_tablespace	The name of the database tablespace to use for this field's index, if this field is indexed. The default is the project's DEFAULT_INDEX_TABLESPACE setting, if set, or the db_tablespace of the model, if any. If the backend doesn't support tablespaces for indexes, this option is ignored.
Default	The default value for the field. This can be a value or a callable object. If callable it will be called every time a new object is created.
help_text	Extra "help" text to be displayed with the form widget. It's useful for documentation even if your field isn't used on a form.
primary_key	If True, this field is the primary key for the model
editable	If False, the field will not be displayed in the admin or any other ModelForm. They are also skipped during model validation. Default is True.
error_messages	The error_messages argument lets you override the default messages that the field will raise. Pass in a dictionary with keys matching the error messages you want to override.
verbose_name	A human-readable name for the field. If the verbose name isn't given, Django will automatically create it using the field's attribute name, converting underscores to spaces.
validators	A list of validators to run for this field. See the validators documentation for more information.
Unique	If True, this field must be unique throughout the table.

<https://docs.djangoproject.com/en/4.2/ref/models/fields/#charfield>

capacity = models.IntegerField(validators=[validate_room_size])

In Django, validators are simple functions (or methods) that are used to check the value of a field against a certain set of rules and raise an error if the value doesn't meet the rules.

Validators are generally used in form fields, but you can also add them to your model fields. A validator can be as simple as a function that checks whether a value is in a specific range or as complex as one that checks whether a string matches a certain regular expression.

validators

```

1 from django.core.exceptions import ValidationError
2
3 def validate_even(value):
4     if value % 2 != 0:
5         raise ValidationError(
6             f'{value} is not an even number',
7             params={'value': value},
8         )

```

This is a simple validator function that checks whether a value is even. If the value is not even, it raises a `ValidationError`.

In this example, the `even_field` must always contain an even number. If you try to save a `MyModel` instance with an odd number in the `even_field`, Django will raise a `ValidationError`.

Django also provides a set of built-in validators that you can use. These are located in `django.core.validators`. Here are some examples:

- **EmailValidator:** Validates whether the value is a valid email address.
- **MinLengthValidator:** Validates whether the length of the value is greater than or equal to the given minimum length.
- **MaxLengthValidator:** Validates whether the length of the value is less than or equal to the given maximum length.
- **MinValueValidator:** Validates whether the value is greater than or equal to the given minimum value.
- **MaxValueValidator:** Validates whether the value is less than or equal to the given maximum value.
- **URLValidator:** Validates whether the value is a valid URL.

```
number = models.IntegerField(validators=[MinValueValidator(0), MaxValueValidator(100)])
```

```
class_room_id = models.AutoField(auto_created = True,  
primary_key = True, serialize = False, verbose_name  
='class_room_id')
```

In Django, **AutoField** is a type of field used to automatically increment an integer value by 1 every time a new instance of the model is created. This type of field is typically used for primary key fields in the model, though it's not mandatory to do so.

It's worth noting that if you don't specify a primary key field for your model, Django will automatically create an AutoField named id, so you don't need to add an AutoField to your model unless you want to change this default behavior.

If you need a non-integer primary key, you can use the UUIDField along with default=uuid.uuid4 and primary_key=True parameters to create a UUID as the primary key.

```
id = models.UUIDField(primary_key=True, default=uuid.uuid4, editable=False)
```

UUID, Universal Unique Identifier, is a python library which helps in generating random objects of 128 bits as ids.

<https://docs.python.org/3/library/uuid.html>

12.10

manually(bulk) populate our classroom tables

We know how to create a form and populate our database like we did in the user part. Just like users it might not be possible to populate the database using forms or it can simply be too time consuming. We can use a python script to populate our tables from the command line and eventually by uploading a file to do a bulk insert.

To make our example easier we will load a csv file:

```
Lab,20,Bio-Tech,BT-255,Middletown  
Lab,20,Tower,TWR-210,Newburgh  
Lab,20,Bio-Tech,BT-357,Middletown  
Classroom,20,Bio-Tech,BT-358,Middletown  
Lab,25,Bio-Tech,BT-121,Middletown  
Lab,20,Bio-Tech,BT-113,Middletown  
Asynchronous,25,TBA,TBA,Middletown  
Asynchronous,25,TBA,TBA,Newburgh  
Classroom,25,Bio-Tech,BT-251,Middletown
```

12.10.1 script to populate database table

populate table command line

```
1 import os
2 import csv
3 import django
4
5 os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'mywebsite.settings')
6 django.setup()
7
8 from users.models import classroom
9
10 if __name__ == '__main__':
11     with open('classroom.csv') as csv_file:
12         csv_reader = csv.reader(csv_file, delimiter=',')
13
14         for row in csv_reader:
15             classroom.objects.create(roomtype=row[0], capacity=row[1],
16                                     location=row[2], room_number=row[3], campus=row[4])
17             print(row)
18
19         """
20         table=classroom.objects.all()
21         print(table[0]._meta.fields)
22         for row in table:
23             print("-"*30)
24             print(row.roomtype)
25             print(row.capacity)
26             print(row.location)
27             print(row.room_number)
28             print(row.campus)
29             print("-"*30)
30
31         """
```

CLASS_ROOM_ID	ROOMTYPE	CAPACITY	LOCATION	ROOM NUMBER	CAMPUS
10	lab	25	Bio-tech	BT-111	Middletown
9	Classroom	25	Bio-Tech	BT-251	Middletown
8	Asynchronous	25	TBA	TBA	Newburgh
7	Asynchronous	25	TBA	TBA	Middletown
6	Lab	20	Bio-Tech	BT-113	Middletown
5	Lab	25	Bio-Tech	BT-121	Middletown
4	Classroom	20	Bio-Tech	BT-358	Middletown
3	Lab	20	Bio-Tech	BT-357	Middletown
2	Lab	20	Tower	TWR-210	Newburgh
1	Lab	20	Bio-Tech	BT-255	Middletown

12.11

create all the other tables and junction tables(ManytoMany)

12.11.1 Classes

classroom-model

```

1 class classes(models.Model):
2     INSTRUCTION_METHOD = (
3         ("In-Person", "In-Person"),
4         ("Remote", "Remote"),
5         ("Off-Campus", "Off-Campus"),
6     )
7     CAMPUS = (
8         ("Middletown", "Middletown"),
9         ("Newburgh", "Newburgh"),
10        ("Other", "Other"),
11    )
12    SESSION = (

```

```

13     ("Fall", "Fall"),
14     ("Spring", "Spring"),
15     ("Summer-Full", "Summer-Full"),
16     ("Summer 1", "Summer 1"),
17     ("Summer 2", "Summer 2"),
18     ("Winter", "Winter"),
19     ("Other", "Other"),
20 )
21 class_room_id = models.ForeignKey(classroom, on_delete=models.DO_NOTHING,
22     ↴ )
22 CRN = models.AutoField(auto_created = True,
23     primary_key = True,
24     serialize = False,
25     verbose_name = 'class_room_id')
26 teacher_id = models.ForeignKey(CustomUser, on_delete=models.DO_NOTHING)
27 description = models.TextField()
28 size = models.IntegerField()
29 registered_total = models.IntegerField(default=0)
30 wait_list = models.IntegerField(default=0)
31 time = models.CharField(max_length = 100)
32 credits = models.IntegerField(default=0)
33 day_of_week = models.CharField(max_length = 30)
34 semester = models.CharField(max_length = 30)
35 semester_year = models.IntegerField()
36 semester_session = models.CharField(max_length = 30, choices=SESSION)
37 subject = models.CharField(max_length = 100)
38 instruction_method = models.CharField(max_length = 100, choices=
39     ↴ INSTRUCTION_METHOD, default="In-Person")
40 campus = models.CharField(max_length = 100, choices=CAMPUS, default="",
41     ↴ Middletown")
42 time_start = models.TimeField()
43 time_end = models.TimeField()
44
45 def __str__(self):
46     return self.CRN

```

```

(orange) occc@occc-VirtualBox:/csc108/project1/mywebsite$ python manage.py makemigrations
Migrations for 'users':
  users/migrations/0005_alter_classroom_campus_classes.py
    - Alter field campus on classroom
    - Create model classes
(orange) occc@occc-VirtualBox:/csc108/project1/mywebsite$ vi users/models.py
(orange) occc@occc-VirtualBox:/csc108/project1/mywebsite$ python manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, sessions, users
Running migrations:
  Applying users.0005_alter_classroom_campus_classes... OK
(orange) occc@occc-VirtualBox:/csc108/project1/mywebsite$

```

<https://docs.djangoproject.com/en/4.2/ref/models/fields/#arguments>

student-model

```

1 class student(models.Model):
2     student_id = models.OneToOneField(CustomUser, on_delete=models.DO_NOTHING,
2         ↴ primary_key = True, verbose_name = 'student_id')
3     gpa = models.FloatField()

```

```

4     credits = models.FloatField()
5     major = models.CharField(max_length = 50)
6     courses_taken = models.ManyToManyField(classes)
7
8     def __str__(self):
9         return self.student_id

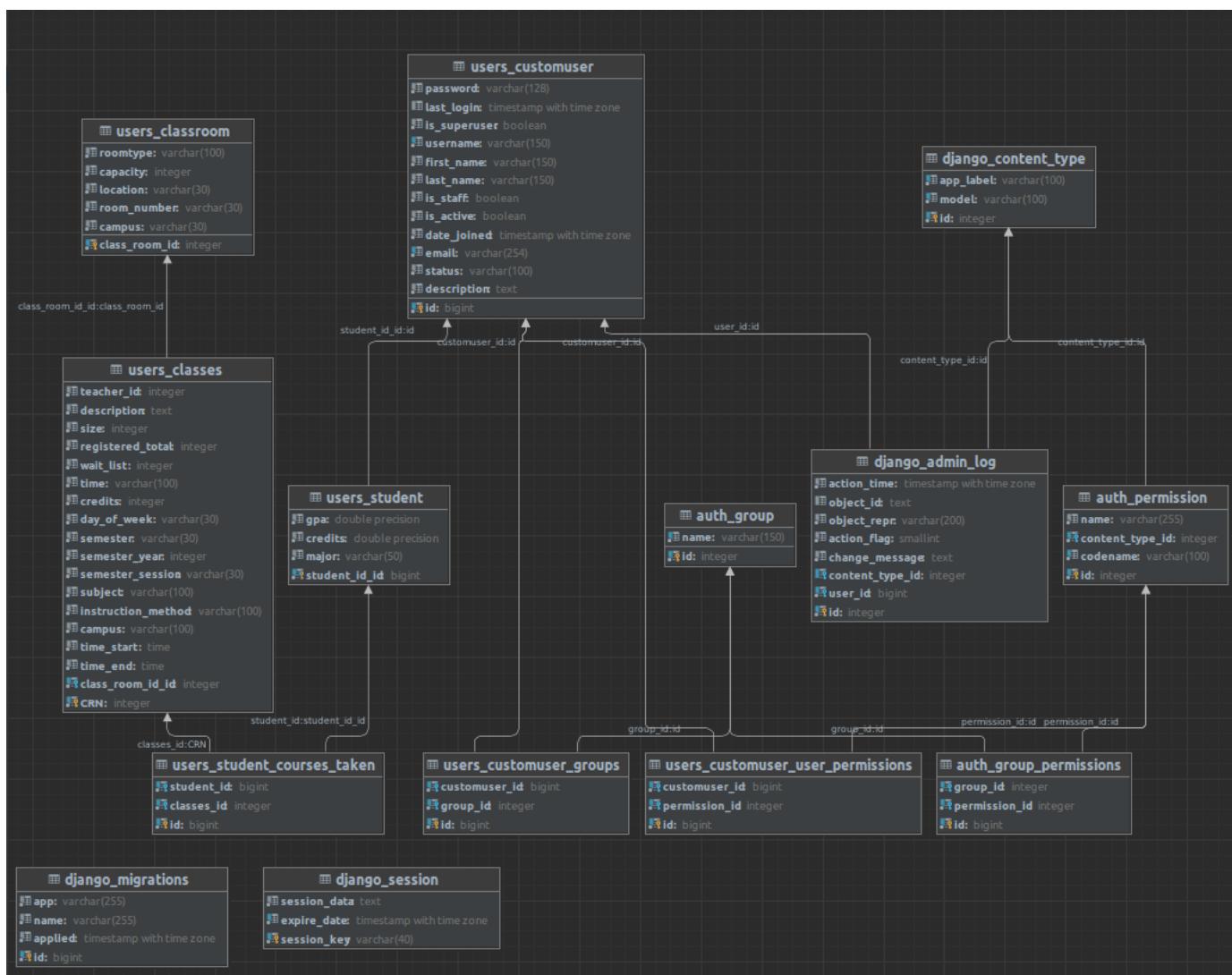
```

```

(orange) occc@occc-VirtualBox:/csc108/project1/mywebsite$ python manage.py makemigrations
Migrations for 'users':
    users/migrations/0006_student.py
        - Create model student
(orange) occc@occc-VirtualBox:/csc108/project1/mywebsite$ python manage.py migrate
Operations to perform:
    Apply all migrations: admin, auth, contenttypes, sessions, users
Running migrations:
    Applying users.0006_student... OK
(orange) occc@occc-VirtualBox:/csc108/project1/mywebsite$ 

```

12.11.2 Current ERD



OneToOneField

In Django, both **ForeignKey** and **OneToOneField** are types of relational fields, which allow you to create relationships between models. However, they establish different types of relationships.

ForeignKey : This establishes a many-to-one relationship. This means that one record in the related model can be connected to multiple records in the model that has the ForeignKey. For example, if you have two models, Author and Book, you can use a ForeignKey in the Book model to link each book to one author, but one author can be linked to many books.

ForeignKey

```

1 class Author(models.Model):
2     name = models.CharField(max_length=200)
3
4 class Book(models.Model):
5     author = models.ForeignKey(Author, on_delete=models.CASCADE)
6     title = models.CharField(max_length=200)
7

```

In this example, a single author can be linked to multiple books, but each book can only have one author.

OneToOneField: This establishes a one-to-one relationship. This means that one record in the related model can only be connected to one record in the model that has the OneToOneField. For example, if you have two models, User and Profile, you can use a OneToOneField in the Profile model to link each profile to one user, and each user can only have one profile.

OneToOneField

```

1 from django.contrib.auth.models import User
2
3 class Profile(models.Model):
4     user = models.OneToOneField(User, on_delete=models.CASCADE)
5     bio = models.TextField()
6

```

In this example, a single user can only be linked to one profile, and vice versa.

In general, use ForeignKey when you need a many-to-one relationship and OneToOneField when you need a one-to-one relationship.

ManyToManyField

The **ManyToManyField** in Django is used to create a many-to-many relationship. This means that a record in a model can be related to any number of records in another model, and each of those records can be related to any number of records in the first model.

A common example is a relationship between Book and Author models, where a book can have multiple authors, and an author can write multiple books.

In our case we used the `onetoonefield` to assure that a student can only have one login account.

ManyToManyField

```
1 class Author(models.Model):
2     name = models.CharField(max_length=200)
3
4 class Book(models.Model):
5     authors = models.ManyToManyField(Author)
6     title = models.CharField(max_length=200)
7
```

In this example, a book can have any number of authors via the `authors` field, and each author can be related to any number of books.

The `ManyToManyField` also provides a number of methods for adding, removing, or clearing related objects. For example, if you have a `Book` instance `book` and an `Author` instance `author`, you can add `author` to the `authors` of `book` with `book.authors.add(author)`. Similarly, you can remove `author` from the `authors` of `book` with `book.authors.remove(author)`, and you can remove all authors of `book` with `book.authors.clear()`.

Note that Django creates an intermediary join table to hold the many-to-many relationship. If you need to associate extra data with the relationship, you can create an intermediary model and use the `through` option on the `ManyToManyField`.

In our case we user `manytomany` to link a student to a class. A class can and almost always will have many students and a students can take several classes.

12.11.3 teacher model

teacher model

```

1 class teacher(models.Model):
2     teacher_id = models.OneToOneField(CustomUser, on_delete=models.CASCADE,
3                                         primary_key=True, verbose_name='teacher_id')
4     title = models.CharField(max_length=50)
5     department = models.CharField(max_length=50, default='NA')
6     courses_teaching = models.ManyToManyField(classes)
7
8     def __str__(self):
9         return self.teacher_id

```

Make sure you do makemigrations and migrate to create the table and join table.

In this case a teacher again can have one and only one login, but can teach multiple classes. Hence the onetoonefield and manytoonefield

register them in the admin.py (optional)

register models in admin using users admin.py

```

1 from django.contrib import admin
2 from django.contrib.auth.admin import UserAdmin
3 from .models import CustomUser, classroom, classes, student, teacher
4
5
6 from django.contrib import admin
7 from .models import CustomUser
8 from .forms import UserRegistrationForm
9 from django.contrib.auth.admin import UserAdmin
10
11
12 # Register your models here.
13 class CustomUserAdmin(UserAdmin):
14     model = CustomUser
15     add_form = UserRegistrationForm
16     fieldsets = (
17         *UserAdmin.fieldsets,
18         (
19             'Other Personal info',
20             {
21                 'fields': (
22                     'status',
23                 )
24             }
25         )

```

```
26     )
27
28
29 admin.site.register(CustomUser,CustomUserAdmin)
30
31 class classroomAdmin(admin.ModelAdmin):
32
33     model = classroom
34     list_display = [f.name for f in classroom._meta.fields]
35
36 admin.site.register(classroom,classroomAdmin)
37
38 class classesAdmin(admin.ModelAdmin):
39
40     model = classes
41     list_display = [f.name for f in classes._meta.fields]
42
43 admin.site.register(classes,classesAdmin)
44
45 class studentAdmin(admin.ModelAdmin):
46
47     model = student
48     list_display = [f.name for f in student._meta.fields]
49
50 admin.site.register(student,studentAdmin)
51
52 class teacherAdmin(admin.ModelAdmin):
53
54     model = teacher
55     list_display = [f.name for f in teacher._meta.fields]
56
57 admin.site.register(teacher,teacherAdmin)
```

USERS	
Classess	+ Add
Classrooms	+ Add
Students	+ Add
Teachers	+ Add
Users	+ Add



Upload data

The easiest to do for starters in our Database model is add a class. the way it is designed right now it has the smallest number of dependencies and we can populate some items with dummy data.

1	Room No.	Teacher	Description	size	credits	Day_of_week	sem_year	session	subject	inst_method	campus	time_start	time_end
2	BT-357	teacher1	Web Development 1	20	4	T,R	2023 Fall	CSC108	In-Person	Middletown	04:00:00 PM	06:50:00 PM	
3	BT-355	teacher1	Comp sci 1	20	4	M,W	2023 Fall	CSC101	In-Person	Middletown	08:00:00 AM	10:50:00 AM	
4	BT-121	teacher1	Scripting	20	3	M,W	2023 Fall	CSC138	In-Person	Middletown	10:00:00 AM	11:50:00 AM	
5	BT-355	teacher1	Scripting	20	3	W	2023 Fall	CSC138	In-Person	Middletown	06:00:00 PM	09:50:00 PM	
6	TWR-210	teacher1	Data com	20	3	M,W	2023 Fall	CIT105	In-Person	Newburg	09:00:00 AM	10:50:00 AM	
7	BT-255	teacher1	Database Fundamentals	20	3	M,W	2023 Fall	CIT225	In-Person	Middletown	01:00:00 PM	02:50:00 PM	

users/form.py

```

1 @login_required
2 def upload_classes(request):
3     """
4         Accepts a csv semicolon(); seperated file in request POST
5         Parses the file and populates the classes table
6         Generates errors response with bad rows that could not meet referential
7             ↴ integrity
8         TODO: Check if we are inserting duplicate data and possibly bad upload ↴
9             ↴ with non csv data
10    """
11    if(request.session['role'] != 'admin'):
12        return HttpResponseRedirect('401 Unauthorized', status=401)
13    else:
14
15        data = {}
16        if request.method == "GET":
17            return render(request, "registrar/upload_classes.html", data)
18        # if not GET, then proceed
19        try:
20            csv_file = request.FILES["csv_file"]
21            #print(csv_file)
22            if not csv_file.name.endswith('.csv'):
23                messages.error(request,'File is not CSV type')
24                return render(request, "registrar/upload_classes.html", ↴
25                                ↴ data)

```

```

23     #if file is too large, return
24     if csv_file.multiple_chunks():
25         messages.error(request,"Uploaded file is too big (%.2f MB). ↴
26             ↴ " % (csv_file.size/(1000*1000),))
27         return render(request, "registrar/upload_classes.html", ↴
28             ↴ data)
29
30
31     lines = file_data.split("\n")
32     #loop over the lines and save them in db. If error , store as ↴
33         ↴ string and then display
34     #make sure to import the model
35     #from users.models import classroom
36
37     error_lines_rooms=[]
38     error_lines_teachers=[]
39     header_line = 0
40     for line in lines:
41
42         if(header_line == 0):
43             header_line = 1
44             continue
45
46         fields = line.split(";");
47         room=fields[0]
48         room_queryset = classroom.objects.filter(room_number=room). ↴
49             ↴ values('class_room_id')
50         if room_queryset.exists() == False:
51             error_lines_rooms.append(line)
52
53             continue
54
55         teacher = fields[1]
56         faculty_queryset = CustomUser.objects.filter(username= ↴
57             ↴ teacher).values('id','status')
58
59         if faculty_queryset.exists():
60             if(faculty_queryset[0]['status'] != 'faculty'):
61                 error_lines_teachers.append(line)
62
63             continue
64         else:
65             error_lines_teachers.append(line)
66
67             continue
68         #convert date time to timefield for saving
69         start_time = fields[11]
70         st_time = start_time.split(' ')
71         st_am_pm = st_time[1]
72         st_hms = st_time[0].split(':')
73         if(st_am_pm == 'PM' and int(st_hms[0]) < 12 ):
74             st_hms[0] = str(int(st_hms[0]) + 12)
end_time = fields[12]

```

```

75         end_temp_time = end_time.split(' ')
76         end_am_pm = end_temp_time[1]
77         end_hms = end_temp_time[0].split(":")
78         if(end_am_pm == 'PM' and int(end_hms[0]) < 12 ):
79             end_hms[0] = str(int(end_hms[0]) + 12)
80             start_time_save = datetime.time(int(st_hms[0]), int(st_hms[
81                 [1]), int(st_hms[2]))
82             end_time_save = datetime.time(int(end_hms[0]), int(end_hms[
83                 [1]), int(end_hms[2]))
84
85             save_object =classes.objects.create(
86                 class_room_id=classroom.objects.get(
87                     class_room_id=room_queryset[0]['class_room_id']),
88                 teacher_id=CustomUser.objects.get(
89                     id=faculty_queryset[0]['id']),
90                 description=fields[2],
91                 size=int(fields[3]),
92                 credits=int(fields[4]),
93                 day_of_week=fields[5],
94                 semester_year=int(fields[6]),
95                 semester_session=fields[7],
96                 subject=fields[7],
97                 instruction_method=fields[8],
98                 campus=fields[9],
99                 time_start=start_time_save,
100                time_end=end_time_save)
101
102                #see if there were any errors
103                if ( (len(error_lines_teachers) > 0) or (len(error_lines_rooms) [
104                    > 0))): #errors are present
105                    context = {"errors_rooms": error_lines_rooms,
106                               "errors_teachers" : error_lines_teachers,
107                               }
108
109                return render(request, 'registrar/upload_classes_errors.' [
110                    html', context)
111            else:
112                return render(request, 'registrar/upload_classes_ok.html', [
113                    context)
114
115        except Exception as e:
116            messages.error(request,"Unable to upload file. "+repr(e))
117            context = {}
118            return render(request, 'registrar/upload_classes.html', context)

```

Upload csv file with all classes

No file selected.

errors in rooms

```
BT-355;teacher1;Comp sci 1;20;4;M,W;2023;Fall;CSC101;In-Person;Middletown;08:00:00 AM;10:50:00 AM
```

```
BT-355;teacher1;Scripting;20;3;W;2023;Fall;CSC138;In-Person;Middletown;06:00:00 PM;09:50:00 PM
```

errors in teachers

```
class_room_id=classroom.objects.get(class_room_id=room_queryset[0]['class_room_id'])
```

In Django, a ForeignKey field represents a one-to-many relationship. It's essentially a link from one model to another. This is how Django understands the relationship between the data in your tables. It's not just storing a simple value, but a reference to another object in the database.

So, when you are setting a ForeignKey field, you need to pass an instance of the related model (an object), not just a simple data type like an integer or a string.

foreign key object example

```
1 class Author(models.Model):
2     name = models.CharField(max_length=100)
3
4 class Book(models.Model):
5     title = models.CharField(max_length=100)
6     author = models.ForeignKey(Author, on_delete=models.CASCADE)
```

In this example, each Book is related to an Author. You can't just set Book.author to a string or an integer, because Book.author isn't just a piece of data. It's a reference to an Author object.

So, to set the author for a Book, you'd do something like this:

foreign key object setting

```
1 some_author = Author.objects.get(name="Author's Name")
2 some_book = Book(title="Book's Title", author=some_author)
3 some_book.save()
```

Here, some_author is an Author object, retrieved from the database using a query. This object is used to set the author for a Book.

By using an object, Django can make sure the relationship is valid and the related Author actually exists. It also allows Django to create a SQL query that inserts a reference to the correct Author in the Book table. When you later retrieve the Book from the database, Django will automatically provide you with

the ability to access related Author data through the author field.

foreign key retrieving

```
1 from django.db import models
2
3 class Author(models.Model):
4     name = models.CharField(max_length=100)
5
6 class Book(models.Model):
7     title = models.CharField(max_length=100)
8     author = models.ForeignKey(Author, on_delete=models.CASCADE)
9
10 # Create an author
11 author = Author.objects.create(name='John Smith')
12
13 # Create a book with author
14 book = Book.objects.create(title='My Book', author=author)
15
16 # Query books and access author name using the ForeignKey relationship
17 books = Book.objects.all()
18 for book in books:
19     print(book.title, book.author.name)
```

The key points:

- The Book model has a ForeignKey field called author that points to the Author model.
- We can query Book objects, then access the related Author using book.author. This will fetch the related object to follow the foreign key relationship.
- The ForeignKey handles the SQL join and making this seem like a normal Python attribute lookup.
- So ForeignKeys allow simple traversal from one model to another via queries.

13.1 > create loading pages for rest of tables

13.1.1 > users

User table

fields in users

```
['password',
'last_login',
'is_superuser',
'groups',
'user_permissions',
'username',
'first_name',
'last_name',
'is_staff',
'is_active',
'date_joined',
'email',
'status',
'description',
'password1',
'password2']
```

users csv file

```
username;first_name;last_name;email;status;description
mkrajca;Miroslav;Krajca;mk@sunnyorange.edu.local;faculty;Professor in compsci dept
akramer;Alessia ;Kramer;ak01@sunnyorange.edu.local;student;student
jglover;James ;Glover;jg01@sunnyorange.edu.local;student;student
cmendez;Catherine ;Mendez;cm01@sunnyorange.edu.local;student;student
rfloyd;Rylan ;Floyd;fr01@sunnyorange.edu.local;admin;registrar
xcombs;Xzavier ;Combs;xc01@sunnyorange.edu.local;student;student
cyang;Cassius ;Yang;cy01@sunnyorange.edu.local;faculty;Professor in compsci dept
kpatel;Kara ;Patel;kp01@sunnyorange.edu.local;student;Professor in compsci dept
nesparza;Noemi ;Esparza;ne01@sunnyorange.edu.local;student;student
mlee;Macy ;Lee;ml01@sunnyorange.edu;student;student
```

Possible issues which we did not take into account last time. Extra spaces been fields.

user upload form

```
1  {% extends "registrar/base.html" %} 
2  {% load static %} 
3  {% block title %} 
4      Upload user list 
5  {% endblock title %}
```

```
6  {% block custom_css %}  
7      <style>  
8          body {  
9              background-color: rgb(163, 161, 160);  
10         }  
11     </style>  
12 {% endblock custom_css %}  
13 {% block content %}  
14     <form action="/registrar/upload_users/"  
15         method="POST"  
16         enctype="multipart/form-data"  
17         class="form-horizontal">  
18     {% csrf_token %}  
19     <div class="form-group">  
20         <label for="csv_file" class="col-md-3 col-sm-3 col-xs-12 control-✓  
21             ↴ label">  
22             <h1>Upload csv file with all users</h1>  
23         </label>  
24         <div class="col-md-8">  
25             <input type="file"  
26                 name="csv_file"  
27                 id="csv_file"  
28                 required="True"  
29                 class="form-control">  
30         </div>  
31     </div>  
32     <div class="form-group">  
33         <div class="col-md-3 col-sm-3 col-xs-12 col-md-offset-3"  
34             style="margin-bottom:10px">  
35             <button class="btn btn-primary">  
36                 <span class="glyphicon glyphicon-upload" style="margin-right:5px;✓  
37                     ↴ "></span>Upload  
38             </button>  
39         </div>  
40     </div>  
41 {% endblock content %}  
42 {% block custom_js %}  
43 {% endblock custom_js %}
```

Upload csv
file with all
users

Browse... No file selected.

Upload

user upload errors messages page

```

1  {% extends "registrar/base.html" %} 
2  {% load static %} 
3  {% load crispy_forms_filters %} 
4  {% load crispy_forms_tags %} 
5  {% block title %} 
6  Upload errors 
7  {% endblock title %} 
8  {% block custom_css %} 
9  {% endblock custom_css %} 
10  {% block content %} 
11  <table class="table table-striped"> 
12    <thead> 
13      <tr> 
14        <th scope="col">errors in users</th> 
15      </tr> 
16    </thead> 
17    <tbody> 
18      {% for e in errors %} 
19        <tr> 
20          <th scope="row">{{ e }}</th> 
21        </tr> 
22      {% endfor %} 
23    </tbody> 
24  </table> 
25  {% endblock content %} 
26  {% block custom_js %} {% endblock custom_js %} 

```

errors in users

mkrajca;Miroslav;Krajca;mk@sunyorange.edu.local;faculty;Professor in compsci dept

IntegrityError('duplicate key value violates unique constraint "users_customuser_username_key"\nDETAIL: Key (username)=(mkrajca) already exists.\n')

user upload view

```

1 @login_required 
2 def upload_users(request): 
3     """ 
4         Accepts a csv semicolon(); separated file in request POST 
5         Parses the file and populates the classes table 
6         Generates errors response with bad rows that could not meet referential 
7             ↴ integrity 
8         TODO: Check if we are inserting duplicate data and possibly bad upload 
9             ↴ with non csv data 
10            """ 
11     if(request.session['role'] != 'admin'): 
12         return HttpResponse('401 Unauthorized', status=401) 
13     else: 
14         error_lines = [] 

```

```
13     data = []
14     if request.method == "GET":
15         return render(request, "registrar/add_users_csv.html", data)
16     # if not GET, then proceed
17     try:
18         csv_file = request.FILES["csv_file"]
19         #print(csv_file)
20         if not csv_file.name.endswith('.csv'):
21             messages.error(request,'File is not CSV type')
22             return render(request, "registrar/add_users_csv.html", ↴
23                           data)
23         #if file is too large, return
24         if csv_file.multiple_chunks():
25             messages.error(request,"Uploaded file is too big (%.2f MB). ↴
26                             ↴ % (csv_file.size/(1000*1000),))")
27             return render(request, "registrar/add_users_csv.html", data)
28         file_data = csv_file.read().decode("utf-8")
29
30
31     lines = file_data.split("\n")
32
33     line_count = 0
34     for line_input in lines:
35
36         if line_count == 0:
37             line_count = 1
38             continue
39         if len(line_input) < 10:
40             continue
41
42         fields = line_input.split(";;")
43         user_name = fields[0].strip()
44         first_name = fields[1].strip()
45         last_name = fields[2].strip()
46         user_email = fields[3].strip()
47         status = fields[4].strip()
48         desc = fields[4].strip()
49
50         try:
51             CustomUser.objects.create(password=make_password(↪
52                                         ↴ user_name+last_name+"123!123!"),
53                                         email=user_email,
54                                         first_name=first_name,
55                                         last_name=last_name,
56                                         status=status,
57                                         description=desc,
58                                         username=user_name,
59                                         )
60         except Exception as e:
61             error_lines.append(line_input)
62             error_lines.append(repr(e))
63             messages.error(request,"Unable to upload file. "+repr(↪
64                           ↴ e))
```

```

65      #see if there were any errors
66      if ( (len(error_lines) > 0) ): #errors are present
67          context = {"errors": error_lines}
68
69          return render(request, 'registrar/add_users_csv_error.✓
70                          ↴ html', context)
71      else:
72          return render(request, 'registrar/add_users_ok.html', ↴
73                          ↴ context)
73      except Exception as e:
74          messages.error(request,"Unable to upload file. "+repr(e))
75
76      context = {}
76      return render(request, 'registrar/add_users_csv.html', context)

```

We should convert this to bulk_load.

The screenshot shows a Django admin interface for managing users. On the left, a sidebar menu under 'USERS' includes 'Classess', 'Classrooms', 'Students', 'Teachers', and 'Users', with 'Users' currently selected. The main area displays a table of 14 user entries:

Action:	USERNAME	EMAIL	FIRST NAME	LAST NAME	STAFF STATUS
<input type="checkbox"/>	admin1	admin1@localhost	admin	admin	X
<input type="checkbox"/>	akramer	ak01@sunnyorange.edu.local	Alessia	Kramer	X
<input type="checkbox"/>	cmendez	cm01@sunnyorange.edu.local	Catherine	Mendez	X
<input type="checkbox"/>	cyang	cy01@sunnyorange.edu.local	Cassius	Yang	X
<input type="checkbox"/>	jglover	jg01@sunnyorange.edu.local	James	Glover	X
<input type="checkbox"/>	kpatel	kp01@sunnyorange.edu.local	Kara	Patel	X
<input type="checkbox"/>	mkrajca	mk@sunnyorange.edu.local	Miroslav	Krajca	X
<input type="checkbox"/>	mlee	ml01@sunnyorange.edu	Macy	Lee	X
<input type="checkbox"/>	nesparza	ne01@sunnyorange.edu.local	Noemi	Esparza	X
<input type="checkbox"/>	occc	occc@localhost			✓
<input type="checkbox"/>	rfloyd	fr01@sunnyorange.edu.local	Rylan	Floyd	X
<input type="checkbox"/>	student1	student1@localhost	jane	Doe	X
<input type="checkbox"/>	teacher1	teacher1@localhost	staff1	staff1	X
<input type="checkbox"/>	xcombs	xc01@sunnyorange.edu.local	Xzavier	Combs	X

14 users

13.2 ➤ teacher csv populate

teacher model

fields in users

```

class teacher(models.Model):
    teacher_id = models.OneToOneField(CustomUser, on_delete=models.CASCADE,
                                      primary_key=True, verbose_name='teacher_id')
    title = models.CharField(max_length=50)
    department = models.CharField(max_length=50, default='NA')
    courses_teaching = models.ManyToManyField(classes)

    def __str__(self):
        return str(self.teacher_id)

```

users csv file

```

title;department;First_name;Last_name;username
Professor;Computer Science and Technology;Giorgianni;Thomas;gthomas
Instructor;Computer Science and Technology;Russell;Hammond;ghammond
Assistant Professor;Computer Science and Technology;Miroslav;Krajca;mkrajca
Department Chair;Computer Science and Technology;Bruce;Roman;broman
Adjunct;Computer Science and Technology;Cassius;Yang;cyang
Associate Professor;Computer Science and Technology;Cartmell;Warrington;cwarrington

```

teacher upload view

```

1 @login_required
2 def upload_teachers(request):
3     """
4         Accepts a csv semicolon(;) seperated file in request POST
5         Parses the file and populates the classes table
6         Generates errors response with bad rows that could not meet referential
7             integrity
8         TODO: Check if we are inserting duplicate data and possibly bad upload
9             with non csv data
10    """
11    if(request.session['role'] != 'admin'):
12        return HttpResponseRedirect('401 Unauthorized', status=401)
13    else:
14        error_lines = []
15        data = {}
16        if request.method == "GET":
17            return render(request, "registrar/add_teachers_csv.html", data)
18        # if not GET, then proceed
19        try:
20            csv_file = request.FILES["csv_file"]
21            #print(csv_file)
22            if not csv_file.name.endswith('.csv'):
23                messages.error(request, 'File is not CSV type')
24                return render(request, "registrar/add_teachers_csv.html", data)
25            if csv_file.multiple_chunks():
26                messages.error(request, "Uploaded file is too big (%.2f MB).")
27                return render(request, "registrar/add_teachers_csv.html", data)
28        
```

```
28     file_data = csv_file.read().decode("utf-8")
29
30
31     lines = file_data.split("\n")
32
33     line_count = 0
34     for line_input in lines:
35
36         if line_count == 0:
37             line_count = 1
38             continue
39         if len(line_input) < 1:
40             continue
41
42         fields = line_input.split(";")
43         title = fields[0].strip()
44         department = fields[1].strip()
45         first_name = fields[2].strip()
46         last_name = fields[3].strip()
47         username = fields[4].strip()
48
49         error_lines = []
50         try:
51             #we have manytomany field in this case connecting
52             #teacher_id to class_id many to many junstions
53             #will have primary keys on both ends of the junction
54             #foreignkey
55             #in order to make this initially work we have to have ↴
56             #    ↴ some dummy placeholder
57             #class or it will need the proper classes in the csv ↴
58             #    ↴ file
59
59         #dummy placeholder class
60         tmp_class = classes.objects.get(description='Place ↴
61             ↴ holder')
62         print(tmp_class.CRN)
63         print(line_input)
64         faculty_queryset = CustomUser.objects.filter(username=↪
65             ↴ username).values('id', 'status')
66         if faculty_queryset.exists():
67             if(faculty_queryset[0]['status'] != 'faculty'):
68                 error_lines.append(line_input)
69                 print("not faculty")
70                 continue
71             else:
72                 error_lines.append(line_input)
73                 print("does not exist")
74                 continue
75             teacher_id_object = CustomUser.objects.get(id=↪
76                 ↴ faculty_queryset[0]['id'])
77             teacher_db = teacher.objects.create(
78                 teacher_id=teacher_id_object,
79                 title=title,
80                 department=department )
81             #create does automatic save
82             #link the two items classes and teacher with the build ↴
83                 ↴ in junction table
```

```

79         teacher_db.courses_teaching.add(tmp_class)
80         #we can later on add other classes
81
82     except Exception as e:
83         print("in Exception")
84         error_lines.append(line_input)
85         error_lines.append(repr(e))
86         print(repr(e))
87         messages.error(request, "Unable to upload file. "+repr(
88                         e))
89
90     #see if there were any errors
91     if ( (len(error_lines) > 0) ): #errors are present
92         context = {"errors": error_lines}
93
94         return render(request, 'registrar/add_users_csv_error.\
95                         html', context)
96     else:
97         return render(request, 'registrar/add_users_ok.html', {})
98
99     except Exception as e:
100        error_lines.append(line_input)
101        error_lines.append(repr(e))
102        messages.error(request, "Unable to upload file. "+repr(e))
103
104    context = {}
105    return render(request, 'registrar/add_teachers_csv.html', context)

```

Another way of writing the raw sql:

```
teacher_db.courses_teaching.add(tmp_class)
```

Courses teaching is a many to many relationship. courses_teaching = models.ManyToManyField(classes)

This links the two tables together by means of a separate join table.

join table select

```
SELECT id, teacher_id, classes_id
  FROM public.users_teacher_courses_teaching;
```

The table is of the form <Django app name> <table where relationship created> <field on which it is created>

	Id [PK] bigint	teacher_id bigint	classes_id integer
1		3	115
2		4	113
3		5	20
4		6	114
5		7	107
6		8	41

sql to retrieve class that is taught

```

1  SELECT users_customuser.first_name,
2    users_customuser.last_name,
3    users_classes.title as "class-title",
4    users_teacher.title as "faculty-title"
5  FROM public.users_customuser,
6    public.users_teacher_courses_teaching,
7    public.users_teacher,
8    public.users_classes
9 WHERE public.users_customuser.username = 'mkrajca'
10   AND public.users_teacher_courses_teaching.teacher_id = public. ↴
     ↴ users_customuser.id
11   AND public.users_teacher_courses_teaching.classes_id = public. ↴
     ↴ users_classes."CRN"
12   AND public.users_teacher.title IN
13     (select users_teacher.title
14      from public.users_teacher,public.users_customuser
15     where public.users_customuser.username = 'mkrajca'
16     and public.users_teacher.teacher_id_id = public.users_customuser. ↴
     ↴ .id);

```

sql to retrieve class that is taught

```

1  SELECT users_customuser.first_name,
2    users_customuser.last_name,
3    users_classes.title as "class-title",
4    users_teacher.title as "faculty-title"
5  FROM public.users_customuser,
6    public.users_teacher_courses_teaching,
7    public.users_teacher,
8    public.users_classes
9 WHERE public.users_customuser.username = 'mkrajca'
10   and public.users_teacher.teacher_id_id = public.users_customuser.id
11   AND public.users_teacher_courses_teaching.teacher_id = public. ↴
     ↴ users_customuser.id
12   AND public.users_teacher_courses_teaching.classes_id = public. ↴
     ↴ users_classes."CRN";

```

	first_name character varying (150)	last_name character varying (150)	class-title character varying (20)	faculty-title character varying (50)
1	Miroslav	Krajca	TBA	Assistant Professor

Change teacher

HISTORY

mkrajca

Teacher_id:    

Title:

Department:

Courses teaching:

- 40
 - 39
 - 38
 - 37
 - 36
 - 35
 - 41**
- 

Hold down "Control", or "Command" on a Mac, to select more than one.

SAVE

Save and add another

Save and continue editing

Delete

Add one more class.

Change teacher

HISTORY

mkrajca

Teacher_id:    

Title:

Department:

Courses teaching:

- 40
- 39
- 38
- 37
- 36**
- 35**
- 41**

Hold down "Control", or "Command" on a Mac, to select more than one.

SAVE

Save and add another

Save and continue editing

Delete

Data Output Messages Explain X Graph Visualiser X Notifications



	first_name character varying (150) 	last_name character varying (150) 	class-title character varying (20) 	faculty-title character varying (50) 
1	Miroslav	Krajca	CSC108	Assistant Professor
2	Miroslav	Krajca	TBA	Assistant Professor

```
public.users_teacher.teacher_id_id
```

When creating a OneToOne relationship in Django, the database column name defaults to `<related_model_name>_id`.

onetooner id

```
1 from django.db import models
2
3 class User(models.Model):
4     name = models.CharField(max_length=100)
5
6 class Profile(models.Model):
7     user = models.OneToOneField(User, on_delete=models.CASCADE)
8     bio = models.TextField()
```

This will create a `profile` table with a column named `user_id` that points to the `id` column of the `user` table.

The column name can also be explicitly specified using the `db_column` parameter:

column name

```
1 user = models.OneToOneField(User, on_delete=models.CASCADE, db_column='↗
    ↳ my_user')
```

Now the column will be named `my_user` instead of the default `user_id`. So in summary, the default naming convention is `<related_model_name>_id` but this can be overridden if desired.

13.3

retrieve classes taught from website Using ORM

faculty class view

```
1 @login_required
2 def my_classes(request):
3     if(request.session['role'] != 'faculty'):
4         return HttpResponseRedirect('401 Unauthorized', status=401)
5     else:
6
7         queryset = teacher.objects.get(teacher_id=request.session['my_id'])
```

```

8     all_classes = queryset.courses_teaching.all()
9
10    context={}
11    context['query'] = all_classes
12
13    return render(request, 'faculty/my_classes.html', context)
14
15 @login_required
16 def all_classes_ajaxprogressive(request):
17     if(request.session['role'] != 'faculty'):
18         return HttpResponse('401 Unauthorized', status=401)
19     else:
20
21         queryset = teacher.objects.get(teacher_id=request.session['my_id'])
22         all_classes = queryset.courses_teaching.all()
23         counter =1
24         json_data= []
25         for cl in all_classes:
26             data_temp={"id":counter,
27                         "CRN":str(cl.CRN),
28                         "class_room_id":cl.class_room_id.roomtype,
29                         "description":cl.description,
30                         "title":cl.title,
31                         "size":cl.size,
32                         "credits":cl.credits,
33                         "day_of_week":cl.day_of_week,
34                         "instruction_method":cl.instruction_method,
35                         "campus":cl.campus,
36                         "time_start":cl.time_start,
37                         "time_end":cl.time_end
38                     }
39             json_data.append(data_temp)
40             counter = counter + 1
41
42 #ret_data = {"data":data}
43 return JsonResponse({"data":json_data},safe = False)

```

faculty classes html

```

1  {% extends "faculty/base.html" %} 
2  {% load static %} 
3  {% load crispy_forms_filters %} 
4  {% load crispy_forms_tags %} 
5  {% block title %} 
6      All users Page 
7  {% endblock title %} 
8  {% block custom_css %} 
9      <link href="{% static 'mainpage/tabulator-master/dist/css/ ↴ 
10          ↴ tabulator_midnight.min.css' %}" 
11          rel="stylesheet"> 
12  {% endblock custom_css %} 
13  {% block content %} 
14      <span class="d-flex justify-content-center align-items-center input- ↴ 
15          ↴ group-text"> 
16          <h1>HTML table version</h1>

```

```
15    </span>
16    <table class="table table-striped">
17        <thead>
18            <tr>
19                <th scope="col">CRN</th>
20                <th scope="col">Classroom</th>
21                <th scope="col">Description</th>
22                <th scope="col">title</th>
23                <th scope="col">size</th>
24                <th scope="col">credits</th>
25                <th scope="col">day of week</th>
26                <th scope="col">instruction mode</th>
27                <th scope="col">campus</th>
28                <th scope="col">time start</th>
29                <th scope="col">time end</th>
30            </tr>
31        </thead>
32        <tbody>
33            {% for class in query %}
34                <tr>
35                    <th scope="row">{{ class.CRN }}</th>
36                    <td>{{ class.class_room_id }}</td>
37                    <td>{{ class.description }}</td>
38                    <td>{{ class.title }}</td>
39                    <td>{{ class.size }}</td>
40                    <td>{{ class.credits }}</td>
41                    <td>{{ class.day_of_week }}</td>
42                    <td>{{ class.instruction_method }}</td>
43                    <td>{{ class.campus }}</td>
44                    <td>{{ class.time_start }}</td>
45                    <td>{{ class.time_end }}</td>
46                </tr>
47            {% endfor %}
48        </tbody>
49    </table>
50    <span class="d-flex justify-content-center align-items-center input-group-text">
51        <h1>Javascript Version</h1>
52    </span>
53    <div id="faculty_classes"></div>
54 {% endblock content %}
55 {% block custom_js %}
56     <script type="text/javascript"
57         src="{% static 'mainpage/tabulator-master/dist/js/tabulator.min.js' %}"></script>
58     <script>
59         //Build Tabulator
60         var table = new Tabulator("#faculty_classes", {
61             height: "200px",
62             layout: "fitColumns",
63             ajaxURL: "/faculty/all_classes_ajaxprogressive",
64             progressiveLoad: "scroll",
65             paginationSize: 20,
66             placeholder: "No Data Set",
67             columns: [
68                 { title: "CRN", field: "CRN",
```

```
70         sorter: "string",
71         width: 200
72     }, {
73         title: "class_room_id",
74         field: "class_room_id",
75         sorter: "number"
76     }, {
77         title: "description",
78         field: "description",
79         sorter: "string"
80     }, {
81         title: "title",
82         field: "title",
83         sorter: "string",
84         hozAlign: "center",
85         width: 100
86     }, {
87         title: "size",
88         field: "size",
89         sorter: "number"
90     }, {
91         title: "credits",
92         field: "credits",
93         sorter: "number",
94         hozAlign: "center"
95     }, {
96         title: "day_of_week",
97         field: "day_of_week",
98         hozAlign: "center",
99         sorter: "string"
100    }, {
101        title: "instruction_method",
102        field: "instruction_method",
103        hozAlign: "center",
104        sorter: "string"
105    }, {
106        title: "campus",
107        field: "campus",
108        hozAlign: "center",
109        sorter: "string"
110    }, {
111        title: "time_start",
112        field: "time_start",
113        hozAlign: "center",
114        sorter: "time"
115    }, {
116        title: "time_end",
117        field: "time_end",
118        hozAlign: "center",
119        sorter: "time"
120    }, ],
121    });
122    </script>
123 {% endblock custom_js %}
```

faculty urls

```

1 from django.urls import path
2
3 from . import views
4
5 urlpatterns = [
6     path("faculty", views.home, name="faculty"),
7     path("faculty/my_classes/",views.my_classes,name="my_classes"),
8     path("faculty/all_classes_ajaxprogressive",views.↪
9         ↴ all_classes_ajaxprogressive,name="all_classes_ajaxprogressive"),
9 ]

```

HTML table version

CRN	Classroom	Description	title	size	credits	day of week	instruction mode	campus	time start	time end
35	BT-357	Web Development 1	CSC108	20	4	T,R	In-Person	Middletown	4 p.m.	6:50 p.m.
41	NA	Place holder	TBA	20	0	M,T,W,R,F	In-Person	Middletown	1:06 a.m.	1:06 a.m.

Javascript Version

CRN	▲ class_ro...	▲ description	▲ title	▲ size	▲ credits	▲ day_of_...	▲ Instructi...	▲ campus	▲ time_start	▲ time_end
35	Lab	Web Develop...	CSC108	20	4	T,R	In-Person	Middletown	16:00:00	18:50:00
41	Classroom	Place holder	TBA	20	0	M,T,W,R,F	In-Person	Middletown	01:06:07	01:06:08

Table JS

using <https://tabulator.info/> for table JS

13.4 > faculty loading

Loading is when a teacher is assigned classes to teach.

We will need the CRN's for the classes

classes table desc

Column	Type	Default	Table "public.users_classes"s	Collation	Nullable
--------	------	---------	-------------------------------	-----------	----------

--	--	--	--	--	--

CRN	integer ↳ generated by default as identity		not null ↵
teacher_id_id	bigint		not null
description	text		not null
size	integer		not null
registered_total	integer		not null
wait_list	integer		not null
credits	integer		not null
day_of_week	character varying(30)		not null
semester_year	integer		not null
semester_session	character varying(30)		not null
subject	character varying(100)		not null
instruction_method	character varying(100)		not null
campus	character varying(100)		not null
time_start	time without time zone		not null
time_end	time without time zone		not null
class_room_id_id	integer		not null
title	character varying(20)		not null

current semester classes

```
orange=> select "CRN",description,title from users_classes where ↵
    ↳ semester_year=2023 and semester_session='Fall';
```

CRN	description	title
53	Unix/Linux	CIT117
52	Networking1	CIT116
51	Computer Hardware/Software	CIT112
50	HTML	CIT111
49	Computer Literacy	CIT100
48	Computer Literacy	CIT100
47	Computer Literacy	CIT100
40	Database Fundamentals	CIT225
39	Data com	CIT105
38	Scripting	CSC138
37	Scripting	CSC138
36	Comp sci 1	CSC101
35	Web Development 1	CSC108
46	Computer Forensics	CFR221
56	Computer applications and graphics	EET110
55	Info Sec	CSS223
54	System Analysis	CIT218

(17 rows)

orange=>

loading csv

CRN	description	title	teacher
53	Unix/Linux	CIT117	mkrajca
52	Networking1	CIT116	mkrajca
51	Computer Hardware/Software	CIT112	cwarrington
50	HTML	CIT111	broman
49	Computer Literacy	CIT100	gthomas

48	Computer Literacy	CIT100	broman
47	Computer Literacy	CIT100	cwarrington
40	Database Fundamentals	CIT225	rhammond
39	Data com	CIT105	broman
38	Scripting	CSC138	mkrajca
37	Scripting	CSC138	mkrajca
36	Comp sci 1	CSC101	mkrajca
35	Web Development 1	CSC108	mkrajca
46	Computer Forensics	CFR221	gthomas
56	Computer applications and graphics	EET110	cyang
55	Info Sec	CSS223	cwarrington
54	System Analysis	CIT218	rhammond

HTML table version

CRN	Classroom	Description	title	size	credits	day of week	instruction mode	campus	time start	time end
41	NA	Place holder	TBA	20	0	M,T,W,R,F	In-Person	Middletown	1:06 a.m.	1:06 a.m.
35	BT-357	Web Development 1	CSC108	20	4	T,R	In-Person	Middletown	4 p.m.	6:50 p.m.
53	BT-121	Unix/Linux	CIT117	20	3	M,W	In-Person	Middletown	8 a.m.	8 a.m.
52	BT-255	Networking1	CIT116	20	4	M,W	In-Person	Middletown	6 p.m.	8:50 p.m.
38	BT-355	Scripting	CSC138	20	3	W	In-Person	Middletown	6 p.m.	9:50 p.m.
37	BT-121	Scripting	CSC138	20	3	M,W	In-Person	Middletown	10 a.m.	11:50 a.m.
36	BT-355	Comp sci 1	CSC101	20	4	M,W	In-Person	Middletown	8 a.m.	10:50 a.m.

Javascript Version

CRN	class_ro...	description	title	size	credits	day_of...	instructi...	campus	time_start	time_end
35	Lab	Web Develop...	CSC108	20	4	T,R	In-Person	Middletown	16:00:00	18:50:00
53	Lab	Unix/Linux	CIT117	20	3	M,W	In-Person	Middletown	08:00:00	08:00:00
52	Lab	Networking1	CIT116	20	4	M,W	In-Person	Middletown	18:00:00	20:50:00
38	lab	Scripting	CSC138	20	3	W	In-Person	Middletown	18:00:00	21:50:00
37	Lab	Scripting	CSC138	20	3	M,W	In-Person	Middletown	10:00:00	11:50:00
36	lab	Comp sci 1	CSC101	20	4	M,W	In-Person	Middletown	08:00:00	10:50:00

facultyloading view

```

1 @login_required
2 def upload_faculty_loading(request):
3     """
4         Accepts a csv semicolon(); separated file in request POST
5         Parses the file and populates the classes table
6         Generates errors response with bad rows that could not meet referential
7             ↴ integrity
8         TODO: Check if we are inserting duplicate data and possibly bad upload ↴
9             ↴ with non csv data
10    """
11
12    if(request.session['role'] != 'admin'):
13        return HttpResponse('401 Unauthorized', status=401)
14    else:
15        error_lines = []
16        data = {}
17        if request.method == "GET":

```

```
16         return render(request, "registrar/teacher_loading.html", data)
17 # if not GET, then proceed
18 try:
19     csv_file = request.FILES["csv_file"]
20     #print(csv_file)
21     if not csv_file.name.endswith('.csv'):
22         messages.error(request,'File is not CSV type')
23         return render(request, "registrar/teacher_loading.html", ↴
24                         data)
25     #if file is too large, return
26     if csv_file.multiple_chunks():
27         messages.error(request,"Uploaded file is too big (%.2f MB). ↴
28                         %.2f" % (csv_file.size/(1000*1000),))
29         return render(request, "registrar/teacher_loading.html", ↴
30                         data)
31
32     file_data = csv_file.read().decode("utf-8")
33     #print(file_data)
34
35     lines = file_data.split("\n")
36
37     line_count = 0
38     for line_input in lines:
39         #print(line_input)
40         if line_count == 0:
41             line_count = 1
42             continue
43         if len(line_input) < 1:
44             continue
45         fields = line_input.split(";;")
46         #get the username and make sure that they are faculty
47         try:
48             teacher_username = fields[3].strip()
49             #print(teacher_username)
50             faculty_queryset = CustomUser.objects.filter(username=↪
51                                         teacher_username).values('id','status')
52             if faculty_queryset.exists():
53                 if(faculty_queryset[0]['status'] != 'faculty'):
54                     error_lines.append(line_input)
55                     #print("not faculty")
56                     continue
57                 else:
58                     error_lines.append(line_input)
59                     #print("does not exist")
60                     continue
61             teacher_id_object = CustomUser.objects.get(id=↪
62                                         faculty_queryset[0]['id'])
63             teacher_db = teacher.objects.get(teacher_id=↪
64                                         teacher_id_object.id)
65             #get the class object
66             classes_object = classes.objects.get(CRN=fields[0].strip())
67             #print(classes_object.title)
68             #update the classes with the new teacher
69             classes_object.teacher_id = teacher_id_object
```

```
66         #teacher_id is a foreign filed to customuser
67         classes_object.save()
68         teacher_db.courses_teaching.add(classes_object)
69     except Exception as e:
70         print("in Exception")
71         print(line_input)
72         error_lines.append(line_input)
73         error_lines.append(repr(e))
74         print(repr(e))
75         messages.error(request,"Unable to upload file. "+repr(
76             ↴ e))
77
78     #see if there were any errors
79     if ( (len(error_lines) > 0) ): #errors are present
80         context = {"errors": error_lines}
81
82         return render(request, 'registrar/add_users_csv_error.'+
83             ↴ 'html', context)
83     else:
84         return render(request, 'registrar/add_users_ok.html', {})
85
86     except Exception as e:
87         error_lines.append(line_input)
88         error_lines.append(repr(e))
89         messages.error(request,"Unable to upload file. "+repr(e))
90
91     context = {}
92     return render(request, 'registrar/teacher_loading.html', context)
```

Action: Go 0 of 18 selected

<input type="checkbox"/>	CLASS_ROOM_ID	CLASS ROOM ID	TEACHER ID	DESCRIPTION	TITLE	SIZE	REGISTERED	TOTAL	WA
<input type="checkbox"/>	56	TWR-210	cyang	Computer applications and graphics	EET110	20	0	0	0
<input type="checkbox"/>	55	BT-357	cwarrington	Info Sec	CSS223	20	0	0	0
<input type="checkbox"/>	54	BT-113	rhammond	System Analysis	CIT218	20	0	0	0
<input type="checkbox"/>	53	BT-121	mkrajca	Unix/Linux	CIT117	20	0	0	0
<input type="checkbox"/>	52	BT-255	mkrajca	Networking1	CIT116	20	0	0	0
<input type="checkbox"/>	51	BT-253	cwarrington	Computer Hardware/Software	CIT112	20	0	0	0
<input type="checkbox"/>	50	BT-121	broman	HTML	CIT111	20	0	0	0
<input type="checkbox"/>	49	BT-253	gthomas	Computer Literacy	CIT100	20	0	0	0
<input type="checkbox"/>	48	BT-253	broman	Computer Literacy	CIT100	20	0	0	0
<input type="checkbox"/>	47	TBA	cwarrington	Computer Literacy	CIT100	20	0	0	0
<input type="checkbox"/>	46	BT-113	gthomas	Computer Forensics	CFR221	20	0	0	0
<input type="checkbox"/>	41	NA	TBA	Place holder	TBA	20	0	0	0
<input type="checkbox"/>	40	BT-255	rhammond	Database Fundamentals	CIT225	20	0	0	0
<input type="checkbox"/>	39	TWR-210	broman	Data com	CIT105	20	0	0	0
<input type="checkbox"/>	38	BT-355	mkrajca	Scripting	CSC138	20	0	0	0
<input type="checkbox"/>	37	BT-121	mkrajca	Scripting	CSC138	20	0	0	0
<input type="checkbox"/>	36	BT-355	mkrajca	Comp sci 1	CSC101	20	0	0	0
<input type="checkbox"/>	35	BT-357	mkrajca	Web Development 1	CSC108	20	0	0	0

13.5 student registration

student register html

```

1  {% extends "students/base.html" %} 
2  {% load static %} 
3  {% load crispy_forms_filters %} 
4  {% load crispy_forms_tags %} 
5  {% block title %} 
6      Student register 
7  {% endblock title %} 
8  {% block custom_css %} 
9      <link rel="stylesheet" 
10         type="text/css" 
11         href="{% static 'mainpage/jquery-ui-1.13.2/jquery-ui.css' %}" /> 
12      <link rel="stylesheet" 
13         href="{% static 'mainpage/grid-3.5.1/pqgrid.min.css' %}" /> 
14      <link rel="stylesheet" 
15         href="{% static 'mainpage/grid-3.5.1/themes/steelblue/pqgrid.css' %}" /> 
16  {% endblock custom_css %}
```

```
17  {% block content %}                                ↴
18    <span class="d-flex justify-content-center align-items-center input-group" style="margin: auto;">
19      <span class="text" style="flex-grow: 1;">Register for classes</span>
20    </span>
21    <div id="grid_editing" style="margin: auto;"></div>
22  {% endblock content %}
23  {% block custom_js %}
24    <script src="{% static 'mainpage/jquery-ui-1.13.2/jquery-ui.min.js' %}"></script>
25    <script src="{% static 'mainpage/grid-3.5.1/pqgrid.min.js' %}"></script>
26  <script>
27    $(function() {
28      var interval;
29      let csrfToken = '{{ csrf_token }}'
30
31      function saveChanges() {
32
33        /**
34         * 1. if there is no active ajax request.
35         * 2. there is no ongoing editing in the grid.
36         * 3. grid is dirty.
37         * 4. all changes are valid.
38       */
39
40      if (!$.active && !grid.getEditCell().$cell && grid.isDirty() && !grid.isEditing() && !grid.isAdding() && !grid.isDeleting()) {
41        $.ajax({
42          allowInvalid: true
43        }).valid() {
44
45          var changes = grid.getChanges({
46            format: 'byVal'
47          });
48          $.ajax({
49            url: "/students/save_registration/",
50            data: {
51              list: JSON.stringify({
52                updateList: changes.updateList,
53                addList: changes.addList,
54                deleteList: changes.deleteList
55              })
56            },
57            dataType: "text",
58            type: "POST",
59            headers: {
60              'X-CSRFToken': csrfToken
61            },
62            mode: 'same-origin',
63            contentType: 'application/json; charset=utf-8',
64            async: true,
65            beforeSend: function(jqXHR, settings) {
66              settings.data = JSON.stringify({
67                updateList: changes.updateList,
68                addList: changes.addList,
69                deleteList: changes.deleteList
70              });
71              grid.option("strLoading", "Saving..");
72            }
73          }).done(function(response) {
74            if (response === "ok") {
75              grid.reload();
76            } else {
77              alert("An error occurred while saving the registration.");
78            }
79          });
80        }
81      }
82    }
83  
```

```
71         grid.showLoading();
72
73     },
74     success: function(changes) {
75
76         //commit the changes.
77         grid.commit({
78             type: 'add',
79             rows: changes.addList
80         });
81         grid.commit({
82             type: 'update',
83             rows: changes.updateList
84         });
85         grid.commit({
86             type: 'delete',
87             rows: changes.deleteList
88         });
89     },
90     complete: function() {
91
92         grid.hideLoading();
93         grid.option("strLoading", $.paramquery.pqGrid.<-->
94             defaults.strLoading);
95     });
96 }
97 }
98 }
99 }
100
101 //save changes from a timer.
102 //interval = setInterval(saveChanges, 1000);
103
104 var obj = {
105     hwrap: false,
106     resizable: true,
107     groupModel: {
108         on: true
109     },
110     virtualX: true,
111     virtualY: true,
112     filterModel: {
113         on: true,
114         mode: "AND",
115         header: true
116     },
117     rowBorders: false,
118     virtualX: true,
119     trackModel: {
120         on: true
121     }, //to turn on the track changes.
122
123     toolbar: {
124         items: [
125             type: 'separator'
126         ],
127     }
128 }
```

```
127          type: 'button',
128          icon: 'ui-icon-disk',
129          label: "register",
130          cls: 'changes',
131          listener: saveChanges,
132          options: {
133              disabled: true
134          }
135      ],
136 },
137 scrollModel: {
138     autoFit: true
139 },
140 historyModel: {
141     checkEditableAdd: true
142 },
143 editor: {
144     select: true
145 },
146 title: "<b>register</b>",
147 change: function(evt, ui) {
148     //saveChanges can also be called from change event.
149 },
150 destroy: function() {
151     //clear the interval upon destroy.
152     clearInterval(interval);
153 },
154
155 colModel: [
156     {
157         title: "CRN",
158         dataType: "id",
159         dataIndx: "CRN",
160         editable: false,
161         width: 80
162     },
163     {
164         title: "classroom",
165         width: 165,
166         dataType: "string",
167         dataIndx: "classroom"
168     },
169     {
170         title: "teacher",
171         width: 140,
172         dataType: "string",
173         dataIndx: "teacher"
174     },
175     {
176         title: "description",
177         width: 100,
178         dataType: "string",
179         dataIndx: "description"
180     },
181     {
182         title: "title",
183         width: 100,
184         dataType: "string",
185         dataIndx: "title",
186         filter: {
```

```
184             type: 'textbox',
185             condition: "begin",
186             listeners: ['keyup']
187         }
188     },
189     {
190         title: "register",
191         width: 100,
192         dataType: "bool",
193         align: "center",
194         dataIndx: "register",
195         editor: false,
196         type: 'checkbox',
197         validations: [
198             {
199                 type: 'nonEmpty',
200                 msg: "Required"
201             }
202         ],
203         title: "size",
204         width: 100,
205         dataType: "integer",
206         dataIndx: "size"
207     },
208     {
209         title: "credits",
210         width: 100,
211         dataType: "integer",
212         dataIndx: "credits"
213     },
214     {
215         title: "day_of_week",
216         width: 100,
217         dataType: "string",
218         dataIndx: "day_of_week"
219     },
220     {
221         title: "dcampus",
222         width: 100,
223         dataType: "string",
224         dataIndx: "campus"
225     },
226     {
227         title: "instruction_method",
228         width: 100,
229         dataType: "string",
230         dataIndx: "instruction_method"
231     },
232     {
233         title: "time_start",
234         width: 100,
235         dataType: "string",
236         dataIndx: "time_start"
237     },
238     ],
239     //postRenderInterval: -1, //synchronous post render.
240     pageModel: {
241         type: "local",
```

```

241         rPP: 20
242     },
243     dataModel: {
244         dataType: "JSON",
245         location: "remote",
246         recIndx: "id",
247         url: "/students/all_classes_list/", //for ASP.NET
248         //url: "/pro/products.php", //for PHP
249         getData: function(response) {
250             return {
251                 data: response.data
252             };
253         }
254     },
255     //rowClick: function (evt, ui) {
256     //    var OrderID = ui.rowData.OrderID;
257     //    grid_registered.data("orderID", OrderID);
258     //    grgrid_registeredid2.pqGrid("option", "dataModel.url", "↖
259     ↴ /students/registered_classes_list" );
260     //    grgrid_registeredd2.pqGrid("refreshDataAndView");
261     //},
262     load: function(evt, ui) {
263         var grid = this,
264             data = grid.option('dataModel').data;
265
266         grid.widget().pqTooltip(); //attach a tooltip.
267
268         //validate the whole data.
269         grid.isValid({
270             data: data
271         });
272     };
273
274     var grid = pq.grid("#grid_editing", obj);
275
276 });
277 </script>
278 {% endblock custom_js %}

```

student svae registration view

```

1 @login_required
2 def save_registration(request):
3     if(request.session['role'] != 'student'):
4         return HttpResponse('401 Unauthorized', status=401)
5     else:
6
7         if request.method == 'POST':
8             body_unicode = request.body.decode('utf-8')
9
10            dataform = str(body_unicode).strip("'<>() ").replace('\'', ↴
11            '\\'')
12
13            json_data = json.loads(dataform)

```

```

13         for item in json_data['updateList']:
14
15             student_db = student.objects.get(student_id=request.session['my_id'])
16             #get the class object
17             classes_object = classes.objects.get(CRN=item['CRN'])
18
19             student_db.courses_taken.add(classes_object)
20
21
22     return HttpResponse("Got json data")

```

Register for classes

CRN	classroom	teacher	description	title	register	size	credits	day_of_w...	dcampus	instructio...	time_start	time_end
136	BT-355	Krajca	Comp sci 1	CSC101	<input checked="" type="checkbox"/>	20	4	M,W	Middletown	In-Person	08:00:00	10:50:00
235	BT-357	Krajca	Web Development 1	CSC108	<input checked="" type="checkbox"/>	20	4	T,R	Middletown	In-Person	16:00:00	18:50:00
346	BT-113	Thomas	Computer Forensics	CFR221	<input checked="" type="checkbox"/>	20	3	T,R	Middletown	In-Person	18:00:00	19:50:00

Page 1 of 1 | Records per page: 20 | Displaying 1 to 17 of 17 items.

13.5.1 drop classes

student drop classes.html

```

1  {% extends "students/base.html" %}
2  {% load static %}
3  {% load crispy_forms_filters %}
4  {% load crispy_forms_tags %}
5  {% block title %}
6      Student register
7  {% endblock title %}
8  {% block custom_css %}
9      <link rel="stylesheet"
10         type="text/css"
11         href="{% static 'mainpage/jquery-ui-1.13.2/jquery-ui.css' %}" />
12      <link rel="stylesheet"
13         href="{% static 'mainpage/grid-3.5.1/pqgrid.min.css' %}" />
14      <link rel="stylesheet"
15         href="{% static 'mainpage/grid-3.5.1/themes/steelblue/pqgrid.css' %}" />
16  {% endblock custom_css %}
17  {% block content %}

```

```
18 <span class="d-flex justify-content-center align-items-center input-group" style="margin: auto;">
19   <input type="text" value="Drop classes"/>
20 </span>
21 <div id="grid_editing" style="margin: auto;"></div>
22 {% endblock content %}
23 {% block custom_js %}
24   <script src="{% static 'mainpage/jquery-ui-1.13.2/jquery-ui.min.js' %}"></script>
25   <script src="{% static 'mainpage/grid-3.5.1/pqgrid.min.js' %}"></script>
26   <script>
27     $(function() {
28       var interval;
29       let csrfToken = '{{ csrf_token }}'
30
31       function saveChanges() {
32
33         /**
34          1. if there is no active ajax request.
35          2. there is no ongoing editing in the grid.
36          3. grid is dirty.
37          4. all changes are valid.
38        */
39
40        if (!$.active && !grid.getEditCell().$cell && grid.isDirty() && !grid.isLoading()) {
41          && grid.isValidChange({
42            allowInvalid: true
43          }).valid) {
44
45           var changes = grid.getChanges({
46             format: 'byVal'
47           });
48           $.ajax({
49             url: "/students/save_drop_classes/",
50             data: {
51               list: JSON.stringify({
52                 updateList: changes.updateList,
53                 addList: changes.addList,
54                 deleteList: changes.deleteList
55               })
56             },
57             dataType: "text",
58             type: "POST",
59             headers: {
60               'X-CSRFToken': csrfToken
61             },
62             mode: 'same-origin',
63             contentType: 'application/json; charset=utf-8',
64             async: true,
65             beforeSend: function(jqXHR, settings) {
66               settings.data = JSON.stringify({
67                 updateList: changes.updateList,
68                 addList: changes.addList,
69                 deleteList: changes.deleteList
70               });
71             grid.option("strLoading", "Saving..");
72           });
73         }
74       }
75     });
76   
```

```
72         grid.showLoading();
73
74     },
75     success: function(changes) {
76
77         //commit the changes.
78         grid.commit({
79             type: 'add',
80             rows: changes.addList
81         });
82         grid.commit({
83             type: 'update',
84             rows: changes.updateList
85         });
86         grid.commit({
87             type: 'delete',
88             rows: changes.deleteList
89         });
90     },
91     complete: function() {
92
93
94
95         grid.hideLoading();
96         grid.option("strLoading", $.paramquery.pqGrid.<
97             ↴ defaults.strLoading);
98     });
99
100 }
101
102
103
104
105 }
106
107 //save changes from a timer.
108 //interval = setInterval(saveChanges, 1000);
109
110 var obj = {
111     hwrap: false,
112     resizable: true,
113     groupModel: {
114         on: true
115     },
116     virtualX: true,
117     virtualY: true,
118     filterModel: {
119         on: true,
120         mode: "AND",
121         header: true
122     },
123     rowBorders: false,
124     virtualX: true,
125     trackModel: {
126         on: true
127     }, //to turn on the track changes.
```

```
128
129     toolbar: {
130         items: [
131             { type: 'separator' }
132         ],
133         items: [
134             { type: 'button',
135                 icon: 'ui-icon-disk',
136                 label: "drop",
137                 cls: 'changes',
138                 listener: saveChanges,
139                 options: {
140                     disabled: true
141                 }
142             }
143         ],
144         scrollModel: {
145             autoFit: true
146         },
147         historyModel: {
148             checkEditableAdd: true
149         },
150         editor: {
151             select: true
152         },
153         title: "<b>Drop</b>",
154         change: function(evt, ui) {
155             //saveChanges can also be called from change event.
156         },
157         destroy: function() {
158             //clear the interval upon destroy.
159             clearInterval(interval);
160         }
161     },
162     colModel: [
163         { title: "CRN",
164             dataType: "id",
165             dataIndx: "CRN",
166             editable: false,
167             width: 80
168         },
169         {
170             title: "classroom",
171             width: 165,
172             dataType: "string",
173             dataIndx: "classroom"
174         },
175         {
176             title: "teacher",
177             width: 140,
178             dataType: "string",
179             dataIndx: "teacher"
180         },
181         {
182             title: "description",
183             width: 100,
184             dataType: "string",
185             dataIndx: "description"
186         }
187     ]
188 }
```

```
185          title: "title",
186          width: 100,
187          dataType: "string",
188          dataIndx: "title",
189          filter: {
190              type: 'textbox',
191              condition: "begin",
192              listeners: ['keyup']
193          }
194      },
195      {
196          title: "drop",
197          width: 100,
198          dataType: "bool",
199          align: "center",
200          dataIndx: "register",
201          editor: false,
202          type: 'checkbox',
203          validations: [
204              {
205                  type: 'nonEmpty',
206                  msg: "Required"
207              }
208          ],
209      },
210      {
211          title: "size",
212          width: 100,
213          dataType: "integer",
214          dataIndx: "size"
215      },
216      {
217          title: "credits",
218          width: 100,
219          dataType: "integer",
220          dataIndx: "credits"
221      },
222      {
223          title: "day_of_week",
224          width: 100,
225          dataType: "string",
226          dataIndx: "day_of_week"
227      },
228      {
229          title: "dcampus",
230          width: 100,
231          dataType: "string",
232          dataIndx: "campus"
233      },
234      {
235          title: "instruction_method",
236          width: 100,
237          dataType: "string",
238          dataIndx: "instruction_method"
239      },
240      {
241          title: "time_start",
242          width: 100,
243          dataType: "string",
244          dataIndx: "time_start"
245      },
246      {
247          title: "time_end",
248          width: 100,
249          dataType: "string",
250          dataIndx: "time_end"
251      },
```

```

242
243     ],
244     postRenderInterval: -1, //synchronous post render.
245     pageModel: {
246       type: "local",
247       rPP: 20
248     },
249     dataModel: {
250       dataType: "JSON",
251       location: "remote",
252       recIndx: "id",
253       url: "/students/registered_classes_list", //for ASP.NET
254       //url: "/pro/products.php", //for PHP
255       getData: function(response) {
256         return {
257           data: response.data
258         };
259       }
260     },
261     //rowClick: function (evt, ui) {
262     //  var OrderID = ui.rowData.OrderID;
263     //  grid_registered.data("orderID", OrderID);
264     //  grgrid_registeredid2.pqGrid("option", "dataModel.url", "↖
265     //    ↳ /students/registered_classes_list" );
266     //  grgrid_registereddd2.pqGrid("refreshDataAndView");
267     //},
268     load: function(evt, ui) {
269       var grid = this,
270         data = grid.option('dataModel').data;
271
272       grid.widget().pqTooltip(); //attach a tooltip.
273
274       //validate the whole data.
275       grid.isValid({
276         data: data
277       });
278     };
279
280     var grid = pq.grid("#grid_editing", obj);
281     grid.on(true, 'refresh refreshCell refreshRow refreshColumn', ↵
282       ↳ function (evt) {
283         console.log(evt.type);
284       });
285     });
286   </script>
287 {%- endblock custom_js %}

```

student drop classes views

```

1 @login_required
2 def registered_classes_list(request):
3   if(request.session['role'] != 'student'):
4     return HttpResponseRedirect('401 Unauthorized', status=401)

```

```
5     else:
6
7         student_db = student.objects.get(student_id=request.session['my_id'] ↴
8             ↴ [])
9         ct = student_db.courses_taken.all().filter(semester_year=2023, ↴
10            ↴ semester_session='Fall')
11         #print(ct)
12         #student_db.courses_taken.remove(41)
13         counter =1
14         json_data= []
15
16         for cl in ct:
17             data_temp={"id":counter,
18                         "CRN":cl.CRN,
19                         "classroom":str(cl.class_room_id.room_number),
20                         "teacher":cl.teacher_id.last_name,
21                         "description":cl.description,
22                         "title":cl.title,
23                         "size":cl.size,
24                         "credits":cl.credits,
25                         "day_of_week":cl.day_of_week,
26                         "instruction_method":cl.instruction_method,
27                         "campus":cl.campus,
28                         "time_start":cl.time_start,
29                         "time_end":cl.time_end
30             }
31             json_data.append(data_temp)
32             counter = counter + 1
33
34             ret_data ={"data":json_data}
35             return JsonResponse(ret_data,safe = False)
36 @login_required
37 def save_drop_classes(request):
38     if(request.session['role'] != 'student'):
39         return HttpResponse('401 Unauthorized', status=401)
40     else:
41
42         if request.method == 'POST':
43             body_unicode = request.body.decode('utf-8')
44
45             dataform = str(body_unicode).strip("'<>() ").replace('\\', ↴
46             ↴ '\\')
47             student_db = student.objects.get(student_id=request.session[ ↴
48                 ↴ ['my_id']])
49             json_data = json.loads(dataform)
50
51             for item in json_data['updateList']:
52                 print(item)
53                 student_db.courses_taken.remove(item['CRN'])
54                 #get the class object
55                 #classes_object = classes.objects.get(CRN=item['CRN'])
56
57                 #student_db.courses_taken.add(classes_object)
58
59
60             return HttpResponse("Got json data")
```

```

58
59 @login_required
60 def drop_classes(request):
61
62     if(request.session['role'] != 'student'):
63         return HttpResponseRedirect('401 Unauthorized', status=401)
64     else:
65         if request.method == 'GET':
66             return render(request, 'students/drop_classes.html')

```

student urls.py

```

1 from django.urls import path
2
3 from . import views
4
5 urlpatterns = [
6     path("students", views.home, name="students"),
7     path("students/register_classes/", views.register_classes, name="register_classes"),
8     path("students/all_classes_list/", views.all_classes_list, name="all_classes_list"),
9     path("students/save_registration/", views.save_registration, name="save_registration"),
10    path("students/registered_classes_list/", views.registered_classes_list, name="registered_classes_list"),
11    path("students/drop_classes/", views.drop_classes, name="drop_classes"),
12    path("students/save_drop_classes/", views.save_drop_classes, name="save_drop_classes"),
13]
14

```

Drop classes

Drop

Drag a column here to group by that column												
CRN	classroom	teacher	description	title	drop	size	credits	day_of_w...	dcampus	instructio...	time_start	time_end
136	BT-355	Krajca	Comp sci 1	CSC101	<input checked="" type="checkbox"/>	20	4	M,W	Middletown	In-Person	08:00:00	10:50:00
235	BT-357	Krajca	Developme nt 1	CSC108	<input type="checkbox"/>	20	4	T,R	Middletown	In-Person	16:00:00	18:50:00
346	BT-113	Thomas	Computer Forensics	CFR221	<input type="checkbox"/>	20	3	T,R	Middletown	In-Person	18:00:00	19:50:00

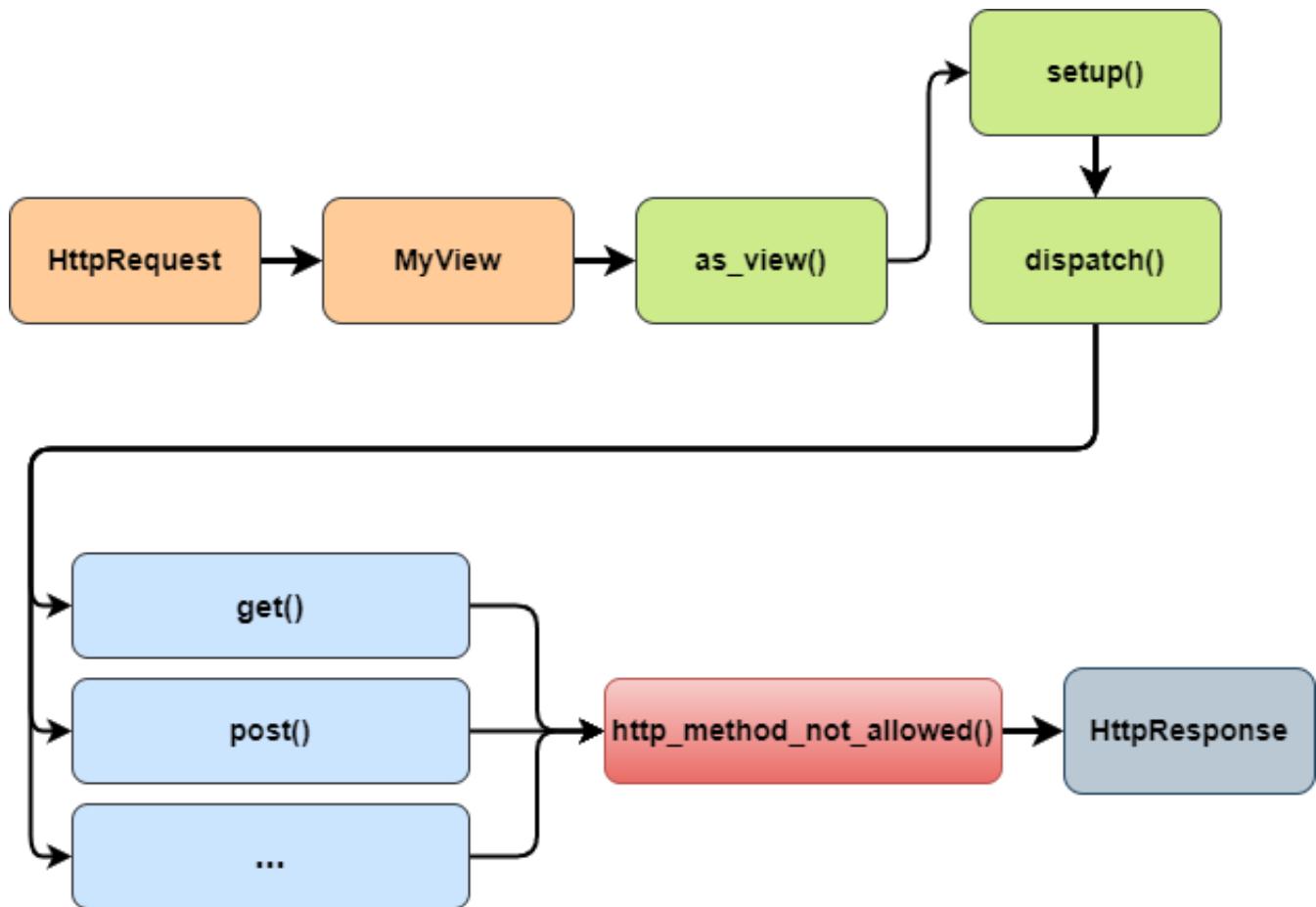
Page 1 of 1 Records per page: 20



Class Based Views

The code flow for CBVs(Class Based Views) is a little more complex because some stuff happens in the background. If we extend the base [View](#) class the following code steps will be executed:

- An `HttpRequest` is routed to `MyView` by the Django URL dispatcher.
- The Django URL dispatcher calls `as_view()` on `MyView`
- `as_view()` invokes `setup()` and `dispatch()`.
- `dispatch()` triggers a method for a specific HTTP method or `http_method_not_allowed()`.
- An `HttpResponse` is returned.



setup method

The `setup()` method in Django class-based views is used to perform initial one-time setup for the view. Some key points:

- It is called by the framework upon initiating the view, before calling other methods like `get()`, `post()` etc.
- Commonly used to perform shared setup logic that needs to happen before handling any HTTP request.
- For example, you may want to check permissions, configure attributes, prefetch common objects etc. before the view handles a request.
- Any attributes attached to the view instance in `setup()` are available to other methods like `get()`, `post()`.
- `setup()` is only called once during the lifetime of a view instance, so it's meant for one-time setup tasks.

- For example:

custom setup in CBV

```

1 class MyView(View):
2     def setup(self, request, *args, **kwargs):
3         self.user = request.user
4         self.category = Category.objects.get(pk=kwargs['category_id'])
5         super().setup(request, *args, **kwargs)

```

original setup in CBV

```

1     def setup(self, request, *args, **kwargs):
2         """Initialize attributes shared by all view methods."""
3         if hasattr(self, "get") and not hasattr(self, "head"):
4             self.head = self.get
5             self.request = request
6             self.args = args
7             self.kwargs = kwargs

```

- Here the user and category are prefetched in `setup()` for easier access later.

So in summary, `setup()` allows keeping shared view initialization logic in one place that gets called before handling requests, rather than repeating it across methods. This makes the view code more modular and DRY.

dispatch method

The `dispatch()` method in Django class-based views is responsible for dispatching requests to the appropriate view method based on the HTTP verb. Here are some key points about it:

- It is called by the framework for every request to the view.
- Checks the HTTP verb (GET, POST, etc) and calls the corresponding method - `get()`, `post()`, etc.
- So `dispatch()` acts like a traffic controller, routing requests to the right handler method.
- Signature: `dispatch(request, *args, **kwargs)`
- `setup()` You can override `dispatch()` to add common pre-processing code

that needs to run before calling view methods.

- For example, authentication checks, permissions, throttling etc.
- The return value of `dispatch()` is used as the response from the view.
- Typical pattern is:

custom dispatch in CBV

```

1 def dispatch(self, request, *args, **kwargs):
2     // add pre-processing code
3
4     response = super().dispatch(request, *args, **kwargs)
5
6     // add post-processing code
7
8     return response

```

original dispatch in CBV

```

1 def dispatch(self, request, *args, **kwargs):
2     # Try to dispatch to the right method; if a method doesn't
3     # exist,
4     # defer to the error handler. Also defer to the error
5     # handler if the
6     # request method isn't on the approved list.
7     if request.method.lower() in self.http_method_names:
8         handler = getattr(
9             self, request.method.lower(), self.
10                http_method_not_allowed
11            )
12     else:
13         handler = self.http_method_not_allowed
14     return handler(request, *args, **kwargs)

```

- This allows hooking in logic before and after calling the standard GET/-POST handling methods.

So in summary, `dispatch()` method handles routing the incoming requests appropriately to the designated view methods like `get()`, `post()` etc. It is a flexible place to put shared preprocessing/postprocessing logic around your view handlers.

http_method_not_allowed

original not_allowed method in CBV

```

1 def http_method_not_allowed(self, request, *args, **kwargs):
2     logger.warning(
3         "Method Not Allowed (%s): %s",
4         request.method,
5         request.path,
6         extra={"status_code": 405, "request": request},
7     )
8     response = HttpResponseRedirect(self._allowed_methods())
9
10    if self.view_is_async:
11
12        async def func():
13            return response
14
15        return func()
16    else:
17        return response

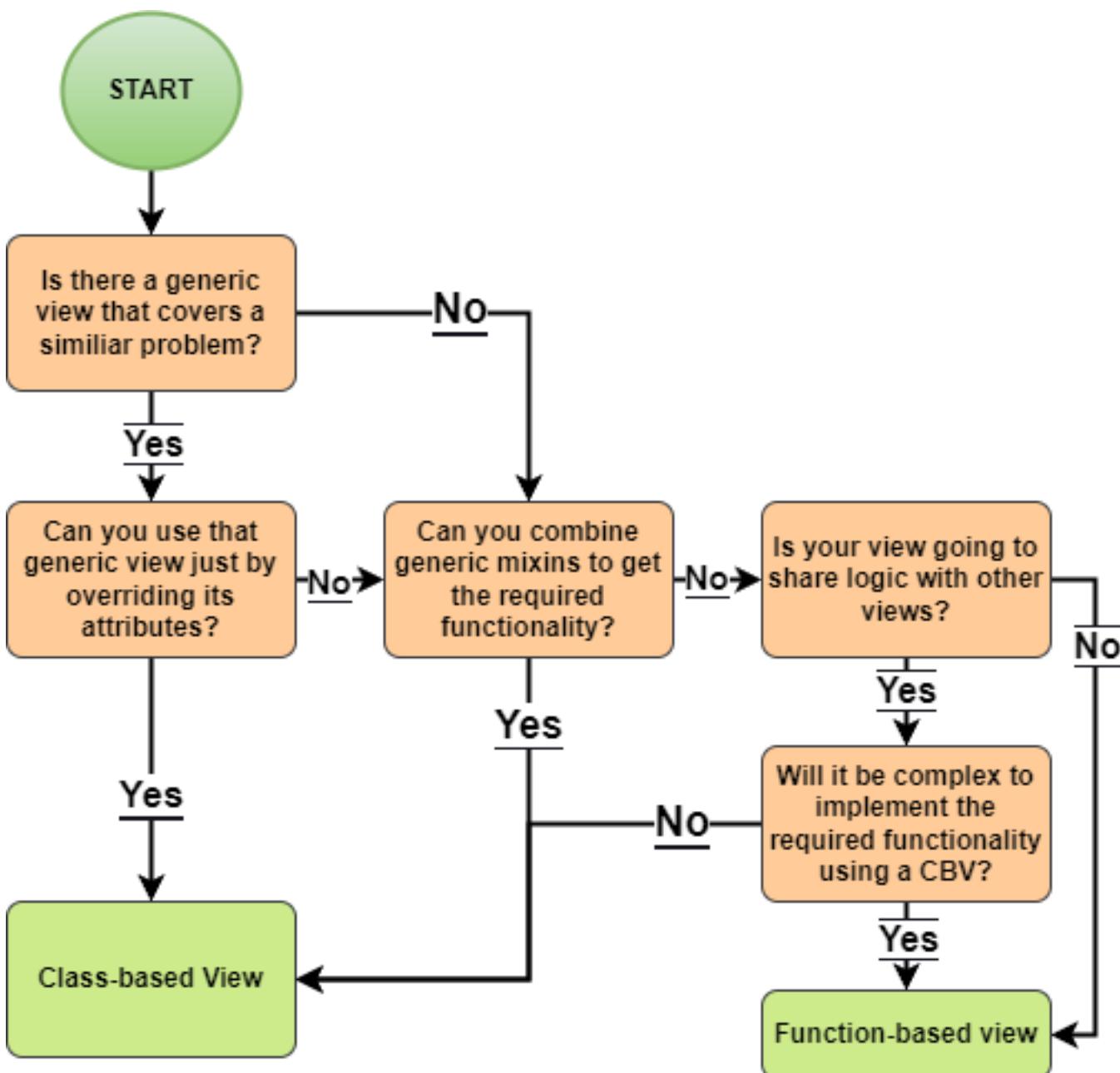
```

The `http_method_not_allowed()` method in Django class-based views is used to generate a 405 Method Not Allowed response. Some key points about it:

- It is called by the `dispatch()` method when a request uses an HTTP verb that is not allowed for that view.
- For example, trying to POST to a view that only allows GET would trigger this.
- It generates a 405 response with an appropriate status code and headers indicating the allowed HTTP methods.
- Where `_allowed_methods()` returns a list of the valid HTTP methods for the view, like `['GET', 'POST']`
- This 405 response communicates to clients that the requested method is invalid and indicates the allowed methods.
- You can override `http_method_not_allowed()` to customize the response. For example, providing a more descriptive error message.
- But it should ideally return a 405 response of some kind to conform to HTTP/RESTful conventions.

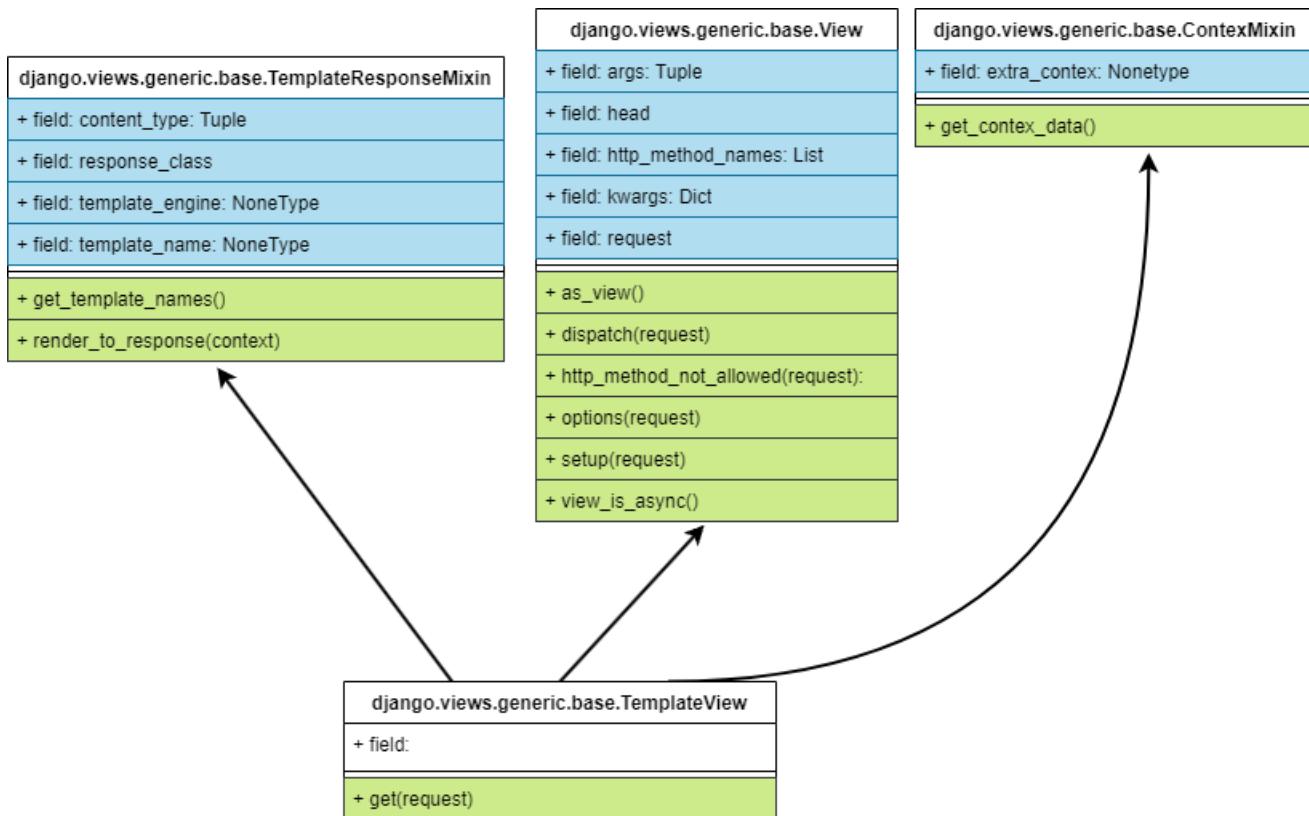
So in summary, `http_method_not_allowed()` generates a standard 405 response when an inappropriate HTTP request is made to the view, allowing clients to detect the error. It helps enforce proper use of the view's allowed HTTP verbs.

14.1 ➤ When to use CBV?



<https://docs.djangoproject.com/en/4.2/topics/class-based-views/>

14.2 ➤ basic templateview UML



14.2.1 templateView

A `TemplateView` class in Django is a generic class-based view for rendering templates. Some key things to know about `TemplateViews`:

- `TemplateView` extends the `View` class in Django. It is designed to quickly render a template without any specific model or form handling.
- `TemplateView` requires specifying a `template_name` attribute that points to the template to render.
- Any extra context data can be provided by overriding the `get_context_data()` method.

get_context_data()

```
1 from django.views.generic import TemplateView
2
3 class MyTemplateView(TemplateView):
4     template_name = 'my_template.html'
5
6     def get_context_data(self, **kwargs):
7         context = super().get_context_data( **kwargs )
8
9         context['my_variable'] = 'Hello, World!'
10        context['my_query'] = teacher.objects.all()
11
12        return context
```

- Common uses of TemplateViews include:

- **Static Pages:** For static pages like 'About Us', 'Contact Us', 'Terms and Conditions', 'Privacy Policy' etc, a TemplateView is ideal since these pages usually don't need to interact with the database. You only need to specify a template_name and Django will render that template.
- **Landing Pages:** If your landing page or homepage has no dynamic content, TemplateView can be used to render these pages. You can use it to display a welcome page, a site map, etc.
- **Custom Error Pages:** Django allows for customization of error pages (like 404, 500, etc.). TemplateView can be used to serve these custom error pages with helpful information and links for the users.
- **HTML Emails:** Sometimes you may need to send an HTML email, which can be treated as a static page. You can create a template for the email content and use TemplateView to render this template into a string that you then use as the body of your email.
- **Prototyping:** During the initial phases of development, you might want to quickly build some placeholder pages. TemplateView can be used to create these pages efficiently before the actual logic is implemented.
- Remember, if you find that your view needs to execute queries or process form data, TemplateView may not be the best choice. In such cases, you would want to use a different view like **ListView**, **DetailView**, **FormView**, etc. which are more suited for those tasks.

- TemplateView will automatically render the template specified in template_name when the view is called.
- No model or form processing is handled. TemplateView just renders the template. Any other logic needs to be handled explicitly in view methods.(but can be created in its methors or get_context_data())
- TemplateView offers a simple way to quickly render a template without needing to create a full custom view class.

So in summary, TemplateView is a generic view that renders a template directly without any specific model or form handling behind the scenes. It's useful for static pages or very simple views.

TemplateView content_type

The content_type attribute on a Django TemplateView controls the Content-Type HTTP header sent with the response when the view is handled.

Specifically:

- By default, TemplateView will set the Content-Type to "text/html" since most TemplateView instances render HTML templates.
- However, you can override content_type to send different Content-Type headers if needed. For example:

```
content_type  
1 from django.views.generic import TemplateView  
2  
3 class MyView(TemplateView):  
4     content_type = 'application/json'  
5     template_name = 'my_template.html'
```

- Here, even though we are rendering an HTML template, we send a Content-Type of application/json.
- This allows building APIs that render templates but send non-HTML content types.
- It also helps ensure the correct Content-Type is sent even if the view returns something other than a rendered template.
- If you don't override content_type, Django will infer the type from the template name extension (.html, .txt, etc) or default to text/html for HTML templates.

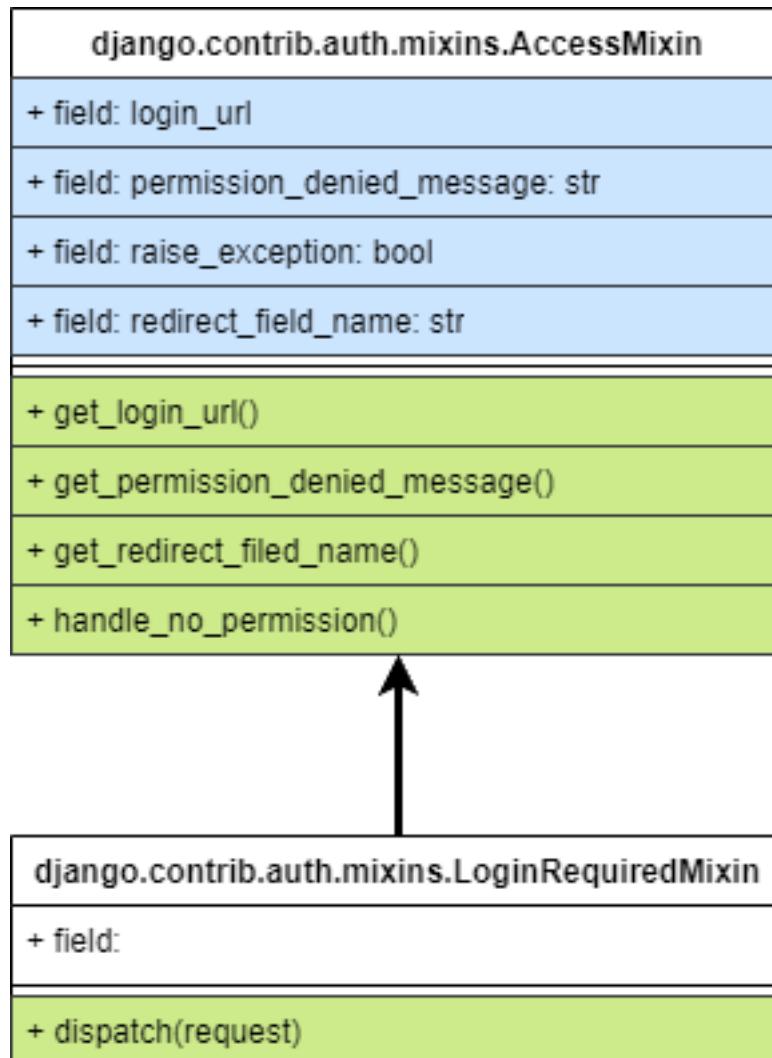
So in summary:

- `content_type` controls the Content-Type HTTP header sent with the response.
- Override it if you need to send a non-standard content type header for a given TemplateView.
- This allows sending JSON/XML/other Content-Types even when rendering HTML templates.

authentication with TemplateView

decorating-the-class

The most simplest way to have authentication is to use the `LoginRequiredMixin`



Here is how you can require a user to be signed in to access a TemplateView in Django:

- Import the LoginRequiredMixin:

```
from django.contrib.auth.mixins import LoginRequiredMixin
```

- Make your TemplateView class inherit from LoginRequiredMixin:

```
class ProtectedView(LoginRequiredMixin, TemplateView):  
    template_name = 'protected.html'
```

- Add the login_url attribute to redirect to the login page:

```
class ProtectedView(LoginRequiredMixin, TemplateView):  
    login_url = '/login/'  
    template_name = 'protected.html'
```

By default, LoginRequiredMixin redirects users to LOGIN_URL defined in your settings.

- Now only authenticated users will be able to access the ProtectedView. Unauthenticated users will be redirected to the /login/ URL.
- You can also use the @method_decorator login_required decorator on function based views:

```
from django.contrib.auth.decorators import login_required  
  
@method_decorator(login_required, name='dispatch')  
class ProtectedView(TemplateView):  
    template_name = 'protected.html'
```

This will again check authentication and redirect to the login page defined in your settings.py if needed before allowing access.

Note: The LoginRequiredMixin should be listed before other classes (like Tem-

plateView) in the order of inheritance. This is because Python's method resolution order (MRO) goes from left to right, so LoginRequiredMixin should be to the left of TemplateView.

14.2.2 Basic list view

create a class in our view.py

```

1  from django.views.generic.list import ListView
2  class teacher_class(ListView):
3      template_name = "registrar/teacher_class.html"
4
5      model = teacher

```

template for class base listview

```

1  {% extends "registrar/base.html" %}
2  {% load static %}
3  {% load crispy_forms_filters %}
4  {% load crispy_forms_tags %}
5  {% block title %}
6  All users Page
7  {% endblock title %}
8  {% block custom_css %}
9  {% endblock custom_css %}
10  {% block content %}

11
12
13 <ul class="list-group">
14     <!-- Iterate over object_list --&gt;
15     {% for object in object_list %}
16         <!-- Display Objects --&gt;
17         &lt;li class="list-group-item list-group-item-primary"&gt;{{ object.title }}&lt;/li&gt;
18         &lt;li class="list-group-item list-group-item-warning"&gt;{{ object.teacher_id.first_name }}&lt;/li&gt;
19         &lt;li class="list-group-item list-group-item-warning"&gt;{{ object.teacher_id.last_name }}&lt;/li&gt;
20         {% for crn in object.courses_teaching.all %}
21             &lt;li class="list-group-item list-group-item-dark"&gt;{{ crn.CRN }}&lt;/li&gt;
22             &lt;li class="list-group-item list-group-item-danger"&gt;
23                 &lt;span class="badge bg-primary rounded-pill"&gt;{{ crn.title }}&lt;/span&gt;
24             &lt;/li&gt;
25
26         {% endfor %}
27         &lt;li class="list-group-item list-group-item-success"&gt;{{ object.teacher_id }}&lt;/li&gt;
28         &lt;hr /&gt;
29         &lt;!-- If object_list is empty --&gt;
30     {% empty %}</pre>

```

```
32      <li>No objects yet.</li>
33      {% endfor %}
34  </ul>
35
36  {% endblock content %}
37  {% block custom_js %}  {% endblock custom_js %}
```

Cassius
Yang
41
TBA
56
EET110
cyang
Associate Professor
Cartmell
Warrington
41
TBA
51
CIT112
47
CIT100
55
CSS223
cwarrington
Assistant Professor
Miroslav
Krajca
41
TBA
35
CSC108
53
CIT117
52
CIT116
38
CSC138
37
CSC138
36
CSC101
mkrajca

We did not even have to specify the template_name it would have been assigned automatically with a name that is the same as the class based view. However the template has to reside in the same app as where the model was created. In our case the user app and this was runned from the registrar app.

14.2.3 pagination on listview

paginator listview

```

1 from django.core.paginator import Paginator, EmptyPage
2 class teacher_pagination(ListView):
3     template_name = "registrar/teacher_class_pagination.html"
4     paginate_by = 2
5     model = teacher

```

paginator url.py

```

1 path('registrar/classbased_teacher/', teacher_class.as_view(), name=' ↴
    ↴ classbased_teacher'),
2 path('registrar/teacher_pagination/', teacher_pagination.as_view(), name=' ↴
    ↴ teacher_paginationr'),

```

paginator template

```

1 {% extends "registrar/base.html" %}
2 {% load static %}
3 {% load crispy_forms_filters %}
4 {% load crispy_forms_tags %}
5 {% block title %}
6     All users Page
7 {% endblock title %}
8 {% block custom_css %}
9 {% endblock custom_css %}
10 {% block content %}
11     <div class='container'>
12         <div class='mt-5 pt-5 ml-5 pl-5'>
13             <ul class="list-group">
14                 <!-- Iterate over object_list -->
15                 {% for object in page_obj %}
16                     <!-- Display Objects -->
17                     <li class="list-group-item list-group-item-primary">{{ object. ↴
                        ↴ title }}</li>
18                     <li class="list-group-item list-group-item-warning">{{ object. ↴
                        ↴ teacher_id.first_name }}</li>
19                     <li class="list-group-item list-group-item-warning">{{ object. ↴
                        ↴ teacher_id.last_name }}</li>
20                 {% for crn in object.courses_teaching.all %}
21                     <li class="list-group-item list-group-item-dark">{{ crn.CRN }}< ↴
                        ↴ /li>
22                     <li class="list-group-item list-group-item-danger">
23                         <span class="badge bg-primary rounded-pill">{{ crn.title }}</span>
24                     </li>
25                 {% endfor %}

```

```
26      <li class="list-group-item list-group-item-success">{{ object.✓
27          ↴ teacher_id }}</li>
28      <hr />
29      <!-- If object_list is empty -->
30      {% empty %}
31          <li>No objects yet.</li>
32      {% endfor %}
33      </ul>
34  </div>
35  <!-- Pagination -->
36  <h1>
37      <div class="pagination">
38          <span class="step-links">
39              {% if page_obj.has_previous %}
40                  <a href="?page=1">&lquo; first</a>
41                  <a href="?page={{ page_obj.previous_page_number }}">previous</a>
42                  ↴
43              {% endif %}
44              <span class="current">Page {{ page_obj.number }} of {{ page_obj.✓
45                  ↴ paginator.num_pages }}.</span>
46              {% if page_obj.has_next %}
47                  <a href="?page={{ page_obj.next_page_number }}">next</a>
48                  <a href="?page={{ page_obj.paginator.num_pages }}">last &raquo;✓
49                  ↴
50              {% endif %}
51          </span>
52      </div>
53      <!-- END Pagination -->
54  </h1>
55  </div>
56  {% endblock content %}
57  {% block custom_js %}
58  {% endblock custom_js %}
```

Professor
Giorgianni
Thomas
41
TBA
49
CIT100
46
CFR221
gthomas
Instructor
Russell
Hammond
41
TBA
40
CIT225
54
CIT218
rhammond

Page 1 of 4. [next](#) [last »](#)

`http://csc108.sunyorange.edu.local:8000/registrar/teacher_pagination/?page=3`

Associate Professor
Cartmell
Warrington
41
TBA
51
CIT112
47
CIT100
55
CSS223
cwarrington

Assistant Professor
Miroslav
Krajca
41
TBA
35
CSC108
53
CIT117
52
CIT116
38
CSC138
37
CSC138
36
CSC101
mkrajca

[« first](#) [previous](#) Page 3 of 4. [next](#) [last »](#)

problem with the above code. What happens if we go past the number of pages or put in a negative value.

```
http://csc108.sunyorange.edu.local:8000/registrar/teacher_pagination/?page=50000
```

Page not found (404)

Invalid page (50000): That page contains no results

Request Method: GET

Request URL: http://csc108.sunyorange.edu.local:8000/registrar/teacher_pagination/?page=50000

Raised by: registrar.views.teacher_pagination

Using the URLconf defined in first_site.urls, Django tried these URL patterns, in this order:

```

1. csc108/
2. admin/
3. login [name='login']
4. logout [name='logout']
5. faculty [name='faculty']
6. faculty/my_classes/ [name='my_classes']
7. faculty/all_classes_ajaxprogressive [name='all_classes_ajaxprogressive']
8. registrar/ [name='registrar']
9. registrar/adduser/ [name='adduser']
10. registrar/allusers/ [name='allusers']
11. registrar/allusersjs/ [name='allusersjs']
12. registrar/allusersjson/ [name='allusersjson']
13. emptypage [name='emptypage']
14. registrar/upload_classes/ [name='upload_classes']
15. registrar/upload_users/ [name='upload_users']
16. registrar/upload_teachers/ [name='upload_teachers']
17. registrar/upload_students/ [name='upload_students']
18. registrar/upload_faculty_loading/ [name='upload_faculty_loading']
19. registrar/db_tests/ [name='db_tests']
20. registrar/orm/teacher_class/ [name='orm_teacher_class_default']
21. registrar/orm/teacher_class/int:id/ [name='orm_teacher_class']
22. registrar/classbased_teacher/ [name='classbased_teacher']
23. registrar/teacher_pagination/ [name='teacher_pagination']

```

The current path, registrar/teacher_pagination/, matched the last one.

You're seeing this error because you have DEBUG = True in your Django settings file. Change that to False, and Django will display a standard 404 page.

paginator custom

```

1 from django.core.paginator import Paginator, EmptyPage
2
3 class teacher_pagination(Paginator):
4     def validate_number(self, number):
5         try:
6             return super().validate_number(number)
7         except EmptyPage:
8             if int(number) > 1:
9                 # return the last page
10                return self.num_pages
11            elif int(number) < 1:
12                # return the first page
13                return 1
14            else:
15                raise
16     class teacher_pagination(ListView):
17         template_name = "registrar/teacher_class_pagination.html"
18         paginate_by = 2
19         model = teacher
20         paginator_class = teacher_pagination # We use our paginator class

```

The custom paginator should prob be in its own file and then imported just to keep views and other code separate.

The screenshot shows a web page with the following structure:

- Header: TBA
- Row 1: TBA
- Row 2: TBA
- Row 3: 40
- Row 4: CIT225 (a blue button)
- Row 5: TBA

[« first](#) [previous](#) Page 4 of 4.

14.2.4 improved pagination

paginator improved

```
1 from django.core.paginator import Paginator, EmptyPage, PageNotAnInteger
2
3 class teacher_pagination(Paginator):
4     def validate_number(self, number):
5         try:
6             return super().validate_number(number)
7         except EmptyPage:
8             if int(number) > 1:
9                 # return the last page
10                return self.num_pages
11            elif int(number) < 1:
12                # return the first page
13                return 1
14            else:
15                raise
16        from django.utils.decorators import method_decorator
17        @method_decorator(login_required(login_url='/login/?next=/registrar/'))
18        ↴ 'teacher_pagination'), name='dispatch')
19    class teacher_pagination(ListView):
20        template_name = "registrar/teacher_class_pagination.html"
21        paginate_by = 2
```

```

1      #model = teacher
2      paginator_class = teacherPaginator # We use our paginator class
3
4      def get(self, request, *args, **kwargs):
5          self.object_list = []
6          if(request.session['role'] != 'admin'):
7              return HttpResponse('401 Unauthorized', status=401)
8          else:
9              context = self.get_context_data()
10             return self.render_to_response(context)
11
12
13      def get_context_data(self, **kwargs):
14          context = super(ListView, self).get_context_data(**kwargs)
15          # Get page number from request,
16          # default to first page
17          default_page = 1
18          page = self.request.GET.get('page', default_page)
19          # Get queryset of items to paginate
20          items = teacher.objects.all().order_by('title')
21          # Paginate items
22          items_per_page = 2
23          paginator = teacherPaginator(items, items_per_page)
24          try:
25              items_page = paginator.page(page)
26          except PageNotAnInteger:
27              items_page = paginator.page(default_page)
28          except EmptyPage:
29              items_page = paginator.page(paginator.num_pages)
30
31          # Provide filtered, paginated library items
32          context["items_page"] = items_page
33
34      return context

```

The @method_decorator decorator in Django is used to decorate class-based views and apply functionality to specific methods of the view.

The dispatch() method is a special method on class-based views that is responsible for dispatching requests to the appropriate handler method (e.g. get(), post(), etc).

So by doing:

```
@method_decorator(login_required, name='dispatch')
```

We are saying - apply the login_required decorator to the dispatch() method of this view.

This means the login_required check will execute on every request to this view, before the specific request method handler is called. So it essentially forces authentication for the entire view, rather than needing to decorate each method individually.

The name argument specifies which method we want to decorate. By specifying 'dispatch', we decorate the dispatch() method.

So in summary:

- dispatch() dispatches requests
- @method_decorator decorates specific view methods
- name='dispatch' decorates the dispatch() method
- This requires login for all handlers in a class-based view

paginator improved html

```

1  {% extends "registrar/base.html" %}          paginator improved html
2  {% load static %}                          1
3  {% load crispy_forms_filters %}            2
4  {% load crispy_forms_tags %}              3
5  {% block title %}                         4
6    All users Page                        5
7  {% endblock title %}                     6
8  {% block custom_css %}                   7
9  {% endblock custom_css %}                 8
10  {% block content %}                    9
11  {% if items_page %}                  10
12    <div class='container'>             11
13      <div class='mt-5 pt-5 ml-5 pl-5'>  12
14        <ul class="list-group">           13
15          <!-- Iterate over object_list --> 14
16          {% for object in items_page %}  15
17            <!-- Display Objects -->       16
18            <li class="list-group-item list-group-item-primary">{{ object. 17
19              ↴ title }}</li>           18
20            <li class="list-group-item list-group-item-warning">{{ object. 19
21              ↴ teacher_id.first_name }}</li>  20
22            <li class="list-group-item list-group-item-warning">{{ object. 21
23              ↴ teacher_id.last_name }}</li>   22
24            {% for crn in object.courses_teaching.all %}  23
25              <li class="list-group-item list-group-item-dark">{{ crn.CRN 24
26                ↴ }}</li>                   25
27              <li class="list-group-item list-group-item-danger">  26
28                <span class="badge bg-primary rounded-pill">{{ crn.title }}</span>  27
29            </li>  28
30          </ul>  29
31      </div>  30
32  </div>  31
33

```

```
26      </li>
27      {% endfor %}
28      <li class="list-group-item list-group-item-success">{{ object.✓
29          ↴ teacher_id }}</li>
30      <hr />
31      <!-- If object_list is empty -->
32      {% empty %}
33      <li>No objects yet.</li>
34      {% endfor %}
35      </ul>
36      </div>
37      <!-- Pagination -->
38      {% if items_page.has_other_pages %}
39          <div class="btn-group" role="group" aria-label="Item pagination">
40              {% if items_page.has_previous %}
41                  <a href="?page={{ items_page.previous_page_number }}" class="btn btn-outline-primary">&laquo; </a>
42              {% endif %}
43              {% for page_number in items_page.paginator.page_range %}
44                  {% if items_page.number == page_number %}
45                      <button class="btn btn-outline-primary active">
46                          <span>{{ page_number }} <span class="sr-only">(current)</span>
47                      </button>
48                  {% else %}
49                      <a href="?page={{ page_number }}" class="btn btn-outline-primary">{{ page_number }}</a>
50                  {% endif %}
51              {% endfor %}
52              {% if items_page.has_next %}
53                  <a href="?page={{ items_page.next_page_number }}" class="btn btn-outline-primary">&raquo; </a>
54              {% endif %}
55          </div>
56          {% endif %}
57      </div>
58      {% else %}
59          <p>No items found.</p>
60      {% endif %}
61  {% endblock content %}
62  {% block custom_js %}
63  {% endblock custom_js %}
```

Adjunct
Cassius
Yang
41
TBA
56
EET110
cyang
Assistant Professor
Miroslav
Krajca
41
TBA
35
CSC108
53
CIT117
52
CIT116
38
CSC138
37
CSC138
36
CSC101
mkrajca

Associate Professor
Cartmell
Warrington
41
TBA
51
CIT112
47
CIT100
55
CSS223
cwarrington
Department Chair
Bruce
Roman
41
TBA
50
CIT111
48
CIT100
39
CIT105
broman

« 1 2 (current) 3 4 »

14.3 ➤ all build in class based views

[class-based-views](#)



Django comes with a build in web server, however this is not good in production. There is no build in modsecurity, https, load-balancing, etc. In order to have these features we have to a real webserver in front of Django that will take care of these items. Most Django sites have nginx in front. We can use apache as well though it is not that often used.

15.1 > install gunicorn

```
occc@occc-VirtualBox:/csc108/project1/orange$ source bin/activate  
(orange) occc@occc-VirtualBox:/csc108/project1/orange$ pip install gunicorn
```

15.2 > install httpie for testing

Although not necessary this utility is handy for testing basic webservers and rest api's from the command line.

<https://httpie.io/cli>

occc@occc-VirtualBox:/csc108/project1/orange\$ source bin/activate
(orange) occc@occc-VirtualBox:/csc108/project1/orange\$



occc@occc-sunyorange

```
occc@occc-VirtualBox:/csc108/project1/orange$ pip install httpie
```

```
(orange) occc@occc-VirtualBox:/csc108/project1/orange$ pip install httpie
Collecting httpie
  Downloading https://pypi.org/simple/httpie/
    127.4/127.4 KB 2.8 MB/s eta 0:00:00
Collecting requests-toolbelt==0.9.1
  Downloading requests_toolbelt-1.0.0-py2.py3-none-any.whl (54 kB)
    54.5/54.5 KB 23.8 MB/s eta 0:00:00
Collecting rich<9.10.0
  Downloading rich-13.4.2-py3-none-any.whl (239 kB)
    239.4/239.4 KB 12.0 MB/s eta 0:00:00
Collecting defusedxml>=0.6.0
  Downloading defusedxml-0.7.1-py2.py3-none-any.whl (25 kB)
Collecting multidict>4.7.0
  Downloading multidict-6.0.4-cp310-cp310-manylinux2014_x86_64.whl (114 kB)
    114.5/114.5 KB 16.4 MB/s eta 0:00:00
Requirement already satisfied: setuptools in ./lib/python3.10/site-packages (from httpie) (59.6.0)
Collecting Pygments>=2.5.2
  Downloading Pygments-2.15.1-py3-none-any.whl (1.1 MB)
    1.1/1.1 MB 51.1 MB/s eta 0:00:00
Collecting charset-normalizer>=2.0.0
  Downloading charset_normalizer-3.1.0-cp310-cp310-manylinux2014_x86_64.whl (199 kB)
    199.3/199.3 KB 57.4 MB/s eta 0:00:00
Collecting requests[socks]>=2.22.0
  Downloading requests-2.31.0-py3-none-any.whl (62 kB)
    62.6/62.6 KB 25.9 MB/s eta 0:00:00
Requirement already satisfied: pip in ./lib/python3.10/site-packages (from httpie) (22.0.2)
Collecting idna>=2.5
  Downloading idna-3.4-py3-none-any.whl (61 kB)
    61.5/61.5 KB 11.4 MB/s eta 0:00:00
Collecting urllib3<3,>=1.21.1
  Downloading urllib3-2.0.3-py3-none-any.whl (123 kB)
    123.6/123.6 KB 24.8 MB/s eta 0:00:00
Collecting certifi>=2017.4.17
  Downloading certifi-2023.5.7-py3-none-any.whl (156 kB)
    157.0/157.0 KB 10.4 MB/s eta 0:00:00
Collecting PySocks!=1.5.7,>=1.5.6
  Downloading PySocks-1.7.1-py3-none-any.whl (16 kB)
Collecting markdown-it-py>=2.2.0
  Downloading markdown_it_py-3.0.0-py3-none-any.whl (87 kB)
    87.5/87.5 KB 31.7 MB/s eta 0:00:00
Collecting mdurl~=0.1
  Downloading mdurl-0.1.2-py3-none-any.whl (10.0 kB)
Installing collected packages: urllib3, PySocks, Pygments, multidict, mdurl, idna, defusedxml, charset-normalizer, certifi, requests, markdown-it-py, rich, requests-toolbelt, httpie
Successfully installed PySocks-1.7.1 Pygments-2.15.1 certifi-2023.5.7 charset-normalizer-3.1.0 httpie-3.2.2 idna-3.4 markdown-it-py-3.0.0 mdurl-0.1.2 multidict-6.0
4 requests-2.31.0 requests-toolbelt-1.0.0 rich-13.4.2 urllib3-2.0.3
(orange) occc@occc-VirtualBox:/csc108/project1/orange$
```

15.3 start django as runserver

```
(orange) occc@occc-VirtualBox:/csc108/project1/django_project/first_lab$ python manage.py check
System check identified no issues (0 silenced).
(orange) occc@occc-VirtualBox:/csc108/project1/django_project/first_lab$ python manage.py runserver &
[1] 13725
(orange) occc@occc-VirtualBox:/csc108/project1/django_project/first_lab$ Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).
June 15, 2023 - 13:41:53
Django version 4.2.2, using settings 'first_lab.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CONTROL-C.

^M
(orange) occc@occc-VirtualBox:/csc108/project1/django_project/first_lab$
```

15.3.1 test that django runserver is listening



occc@occc-sunyorange

```
sudo lsof -n -P -i TCP:8000 -s TCP:LISTEN
```

```
(orange) occc@occc-VirtualBox:/csc108/project1/django_project/first_lab$ sudo lsof -n -P -i TCP:80
00 -s TCP:LISTEN
[sudo] password for occc:
COMMAND   PID USER   FD   TYPE DEVICE SIZE/OFF NODE NAME
python 13726 occc    4u  IPv4 106250      0t0  TCP 127.0.0.1:8000 (LISTEN)
(orange) occc@occc-VirtualBox:/csc108/project1/django_project/first_lab$
```

15.3.2 use httpie to test connection



occc@occc-sunyorange

```
http GET http://csc108.sunyorange.edu.local:8000/csc108/list_example/
```

```
(orange) occc@occc-VirtualBox:/csc108/project1/django_project/first_lab$ http GET http://csc108.sunyorange.edu.local:8000/csc108/list_example/
HTTP/1.1 200 OK
Content-Length: 342
Content-Type: text/html; charset=utf-8
Cross-Origin-Opener-Policy: same-origin
Date: Thu, 15 Jun 2023 13:56:45 GMT
Referrer-Policy: same-origin
Server: WSGIServer/0.2 CPython/3.10.6 [REDACTED]
X-Content-Type-Options: nosniff
X-Frame-Options: DENY[15/Jun/2023 13:56:45] "GET /csc108/list_example/ HTTP/1.1" 200 342

<!DOCTYPE html>
<html lang="en-US">
<head>
  <meta charset="utf-8" >
  <title>template with dictionary</title>
</head>
<body>
  <h1>CSC 108 information</h1>
  <p><strong> Course Name:</strong>Web programming 1</p>
  <p><strong> Course no:</strong> csc108</p>
  <p><strong> Location:</strong> Middletown campus </p>
</body>
</html>

(orange) occc@occc-VirtualBox:/csc108/project1/django_project/first_lab$
```

15.3.3 start gunicorn

stop django runserver.

Start gunicorn and then use a web browser or httpie to make sure it is working.



occc@occc-sunyorange

```
/csc108/project1/orange/bin/gunicorn --bind 0.0.0.0:8000 first_lab.wsgi
```

```
(orange) occc@occc-VirtualBox:/csc108/project1/django_project/first_lab$ /csc108/project1/orange/bin/gunicorn -bind 0.0.0.0:8000 first_lab.wsgi
[2023-06-15 10:25:30 -0400] [14283] [INFO] Starting gunicorn 20.1.0
[2023-06-15 10:25:30 -0400] [14283] [INFO] Listening at: http://0.0.0.0:8000 (14283)
[2023-06-15 10:25:30 -0400] [14283] [INFO] Using worker: sync
[2023-06-15 10:25:30 -0400] [14284] [INFO] Booting worker with pid: 14284
```

```
occc@occc-sunyorange
sudo lsof -n -P -i TCP:8000 -s TCP:LISTEN
```

```
(orange) occc@occc-VirtualBox:/csc108/project1/django_project/first_lab$ sudo lsof -n -P -i TCP:8000 -s TCP:LISTEN
COMMAND    PID USER    FD   TYPE DEVICE SIZE/OFF NODE NAME
gunicorn 14283 occc    5u  IPv4 117468      0t0  TCP *:8000 (LISTEN)
gunicorn 14284 occc    5u  IPv4 117468      0t0  TCP *:8000 (LISTEN)
(orange) occc@occc-VirtualBox:/csc108/project1/django_project/first_lab$
```

```
occc@occc-sunyorange
(orange) occc@occc-VirtualBox:/csc108/project1/django_project/first_lab$ ls
csc108 db.sqlite3 first_lab manage.py
(orange) occc@occc-VirtualBox:/csc108/project1/django_project/first_lab$ gunicorn --bind 0.0.0.0:8000 first_lab.wsgi
[2023-06-13 20:18:30 -0400] [68271] [INFO] Starting gunicorn 20.1.0
[2023-06-13 20:18:30 -0400] [68271] [INFO] Listening at: http://0.0.0.0:8000 (68271)
[2023-06-13 20:18:30 -0400] [68271] [INFO] Using worker: sync
[2023-06-13 20:18:30 -0400] [68272] [INFO] Booting worker with pid: 68272
Not Found: /favicon.ico
Not Found: /favicon.ico
Not Found: /favicon.ico
```



```
(orange) occc@occc-VirtualBox:/csc108/project1/django_project/first_lab$ http GET http://csc108.sunyorange.edu.local:8000/csc108/list_example/
HTTP/1.1 200 OK
Connection: close
Content-Length: 342
Content-Type: text/html; charset=utf-8
Cross-Origin-Opener-Policy: same-origin
Date: Thu, 15 Jun 2023 14:31:22 GMT
Referrer-Policy: same-origin
Server: gunicorn
X-Content-Type-Options: nosniff
X-Frame-Options: DENY

<!DOCTYPE html>
<html lang="en-US">
<head>
  <meta charset="utf-8" >
  <title>template with dictionary</title>
</head>
<body>
  <h1>CSC 108 information</h1>
  <p><strong> Course Name:</strong>Web programming 1</p>
  <p><strong> Course no:</strong> csc108</p>
  <p><strong> Location:</strong> Middletown campus </p>
</body>
</html>

(orange) occc@occc-VirtualBox:/csc108/project1/django_project/first_lab$
```

Stop gunicorn

15.3.4 create a config file for gunicorn

create config directory for gunicorn

```
occc@occc-sunyorange
(orange) occc@occc-VirtualBox:/csc108/project1/django_project/first_lab$ mkdir -p config/gunicorn
```

create log and pid directories for gunicorn

```
occc@occc-sunyorange
(orange) occc@occc-VirtualBox:/csc108/project1/django_project/first_lab$ sudo mkdir -p /var/{log,run}/gunicorn/
(orange) occc@occc-VirtualBox:/csc108/project1/django_project/first_lab$ sudo chown -cR occc:occc /var/{log,run}/gunicorn/
changed ownership of '/var/run/gunicorn/' from root:root to occc:occc
(orange) occc@occc-VirtualBox:/csc108/project1/django_project/first_lab$
```

In project/config/gunicorn directory create a config file. Name it config.py

```
occc@occc-sunyorange
(orange) occc@occc-VirtualBox:/csc108/project1/django_project/first_lab$ more config/gunicorn/config.py
"""
Unicorn *development* config file"""

# Django WSGI application path in pattern MODULE_NAME:VARIABLE_NAME
wsgi_app = "first_lab.wsgi:application"
# The granularity of Error log outputs
loglevel = "debug"
# The number of worker processes for handling requests
workers = 2
# The socket to bind
bind = "0.0.0.0:8000"
# Restart workers when code changes (development only!)
reload = True
# Write access and error info to /var/log
accesslog = "/var/log/gunicorn/access.log"
errorlog = "/var/log/gunicorn/error.log"
# Redirect stdout/stderr to log file
capture_output = True
# PID file so you can easily fetch process ID
pidfile = "/var/run/gunicorn/dev.pid"
# Daemonize the Gunicorn process (detach & enter background)
daemon = True
(orange) occc@occc-VirtualBox:/csc108/project1/django_project/first_lab$
```

start gunicorn with new config gunicorn

```
occc@occc-sunyorange
/csc108/project1/orange/bin/gunicorn -c config/gunicorn/config.py
```

```
occc@occc-sunyorange
(orange) occc@occc-VirtualBox:/csc108/project1/django_project/first_lab$ pwd
/csc108/project1/django_project/first_lab
(orange) occc@occc-VirtualBox:/csc108/project1/django_project/first_lab$ 
(orange) occc@occc-VirtualBox:/csc108/project1/django_project/first_lab$ 
(orange) occc@occc-VirtualBox:/csc108/project1/orange/bin/gunicorn -c config/gunicorn/config.py
```

```
(orange) occc@occc-VirtualBox:/csc108/project1/django_project/first_lab$ tail -f /var/log/gunicorn/error.log
  ciphers: None
  raw_paste_global_conf: []
  strip_header_spaces: False
[2023-06-15 11:15:45 -0400] [14701] [INFO] Starting gunicorn 20.1.0
[2023-06-15 11:15:45 -0400] [14701] [DEBUG] Arbiter booted
[2023-06-15 11:15:45 -0400] [14701] [INFO] Listening at: http://0.0.0.0:8000 (14701)
[2023-06-15 11:15:45 -0400] [14701] [INFO] Using worker: sync
[2023-06-15 11:15:45 -0400] [14702] [INFO] Booting worker with pid: 14702
[2023-06-15 11:15:45 -0400] [14703] [INFO] Booting worker with pid: 14703
[2023-06-15 11:15:45 -0400] [14701] [DEBUG] 2 workers
```

```
(orange) occc@occc-VirtualBox:/csc108/project1/django_project/first_lab$ sudo lsof -n -P -i TCP:8000 -s TCP:LISTEN
COMMAND   PID USER   FD   TYPE DEVICE SIZE/OFF NODE NAME
gunicorn 14701 occc    8u  IPv4 122546      0t0  TCP *:8000 (LISTEN)
gunicorn 14702 occc    8u  IPv4 122546      0t0  TCP *:8000 (LISTEN)
gunicorn 14703 occc    8u  IPv4 122546      0t0  TCP *:8000 (LISTEN)
(orange) occc@occc-VirtualBox:/csc108/project1/django_project/first_lab$
```

```
(orange) occc@occc-VirtualBox:/csc108/project1/django_project/first_lab$ http GET http://csc108.sunyorange.edu.local:8000/csc108/list_example/
HTTP/1.1 200 OK
Connection: close
Content-Length: 342
Content-Type: text/html; charset=utf-8
Cross-Origin-Opener-Policy: same-origin
Date: Thu, 15 Jun 2023 14:31:22 GMT
Referrer-Policy: same-origin
Server: gunicorn
X-Content-Type-Options: nosniff
X-Frame-Options: DENY

<!DOCTYPE html>
<html lang="en-US">
<head>
  <meta charset="utf-8" >
  <title>template with dictionary</title>
</head>
<body>
  <h1>CSC 108 information</h1>
  <p><strong> Course Name:</strong>Web programming 1</p>
  <p><strong> Course no:</strong> csc108</p>
  <p><strong> Location:</strong> Middletown campus </p>
</body>
</html>
```

(orange) occc@occc-VirtualBox:/csc108/project1/django_project/first_lab\$

```
occc@occc-sunyorange
(orange) occc@occc-VirtualBox:/csc108/project1/django_project/first_lab$ which gunicorn
/csc108/project1/orange/bin/gunicorn
(orange) occc@occc-VirtualBox:/csc108/project1/django_project/first_lab$
```

15.3.5 systemd start

use systemd to start gunicorn

<https://docs.gunicorn.org/en/latest/deploy.html>

/etc/systemd/system/gunicorn.service

```
[Unit]
Description=gunicorn daemon
Requires=gunicorn.socket
After=network.target

[Service]
Type=notify
# the specific user that our service will run as
User=occc
Group=www-data
# another option for an even more restricted service is
# DynamicUser=yes
# see http://0pointer.net/blog/dynamic-users-with-systemd.html
RuntimeDirectory=gunicorn
WorkingDirectory=/csc108/project1/django_project/first_lab
#ExecStart=/csc108/project1/orange/bin/gunicorn --bind 0.0.0.0:8000 first_lab.wsgi
ExecStart=/csc108/project1/orange/bin/gunicorn -c config/gunicorn/dev.py

ExecReload=/bin/kill -s HUP $MAINPID
KillMode=mixed
TimeoutStopSec=5
PrivateTmp=true

[Install]
WantedBy=multi-user.target
```

/etc/systemd/system/gunicorn.socket

```
[Unit]
Description=gunicorn socket

[Socket]
ListenStream=/var/run/gunicorn/gunicorn.sock
# Our service won't need permissions for the socket, since it
# inherits the file descriptor by socket activation
# only the nginx daemon will need access to the socket
SocketUser=www-data
# Optionally restrict the socket permissions even more.
# SocketMode=600

[Install]
WantedBy=sockets.target
```



occc@occc-sunyorange

```
root@occc-VirtualBox:~# systemctl daemon-reload
root@occc-VirtualBox:~# systemctl start gunicorn
root@occc-VirtualBox:~# systemctl start gunicorn.socket
root@occc-VirtualBox:~# systemctl enable gunicorn.socket
root@occc-VirtualBox:~# systemctl enable gunicorn
Created symlink /etc/systemd/system/multi-user.target.wants/gunicorn.service →
/etc/systemd/system/gunicorn.service.
root@occc-VirtualBox:~#
```

```
root@occc-VirtualBox:~# systemctl daemon-reload
root@occc-VirtualBox:~# systemctl start gunicorn
root@occc-VirtualBox:~# systemctl start gunicorn.socket
root@occc-VirtualBox:~# systemctl enable gunicorn.socket
root@occc-VirtualBox:~# systemctl enable gunicorn
Created symlink /etc/systemd/system/multi-user.target.wants/gunicorn.service → /
etc/systemd/system/gunicorn.service.
root@occc-VirtualBox:~# █
```

```
root@occc-VirtualBox:~# systemctl status gunicorn.socket
● gunicorn.socket - gunicorn socket
   Loaded: loaded (/etc/systemd/system/gunicorn.socket; enabled; vendor preset: enabled)
   Active: active (running) since Thu 2023-06-15 11:32:43 EDT; 7min ago
     Triggers: ● gunicorn.service
   Listen: /run/gunicorn/gunicorn.sock (Stream)
     Tasks: 0 (limit: 2277)
   Memory: 4.0K
     CPU: 439us
   CGroup: /system.slice/gunicorn.socket

Jun 15 11:32:43 occc-VirtualBox systemd[1]: Starting gunicorn socket...
Jun 15 11:32:43 occc-VirtualBox systemd[1]: Listening on gunicorn socket.
Jun 15 11:33:15 occc-VirtualBox systemd[1]: /etc/systemd/system/gunicorn.socket:5: ListenStream= >
Jun 15 11:33:55 occc-VirtualBox systemd[1]: /etc/systemd/system/gunicorn.socket:5: ListenStream= >
Jun 15 11:34:05 occc-VirtualBox systemd[1]: /etc/systemd/system/gunicorn.socket:5: ListenStream= >
root@occc-VirtualBox:~# systemctl status gunicorn
● gunicorn.service - gunicorn daemon
   Loaded: loaded (/etc/systemd/system/gunicorn.service; enabled; vendor preset: enabled)
   Active: active (running) since Thu 2023-06-15 11:32:46 EDT; 7min ago
     TriggeredBy: ● gunicorn.socket
     Main PID: 14855 (gunicorn)
       Status: "Gunicorn arbiter booted"
     Tasks: 2 (limit: 2277)
   Memory: 42.4M
     CPU: 269ms
   CGroup: /system.slice/gunicorn.service
           └─14855 /csc108/project1/orange/bin/python3 /csc108/project1/orange/bin/gunicorn --b
             ├─14856 /csc108/project1/orange/bin/python3 /csc108/project1/orange/bin/gunicorn --b

Jun 15 11:32:46 occc-VirtualBox systemd[1]: Starting gunicorn daemon...
Jun 15 11:32:46 occc-VirtualBox gunicorn[14855]: [2023-06-15 11:32:46 -0400] [14855] [INFO] Starti>
Jun 15 11:32:46 occc-VirtualBox gunicorn[14855]: [2023-06-15 11:32:46 -0400] [14855] [INFO] Listeni>
Jun 15 11:32:46 occc-VirtualBox gunicorn[14855]: [2023-06-15 11:32:46 -0400] [14855] [INFO] Using i>
Jun 15 11:32:46 occc-VirtualBox systemd[1]: Started gunicorn daemon.
Jun 15 11:32:46 occc-VirtualBox gunicorn[14856]: [2023-06-15 11:32:46 -0400] [14856] [INFO] Booti>
Jun 15 11:33:15 occc-VirtualBox systemd[1]: gunicorn.service: Current command vanished from the u>
root@occc-VirtualBox:~#
```

```
(orange) occc@occc-VirtualBox:csc108/project1/django_project/first_lab$ http GET http://csc108.sunyorange.edu.local:8000/csc108/list_example/
HTTP/1.1 200 OK
Connection: close
Content-Length: 342
Content-Type: text/html; charset=utf-8
Cross-Origin-Opener-Policy: same-origin
Date: Thu, 15 Jun 2023 15:45:09 GMT
Referrer-Policy: same-origin
Server: gunicorn
X-Content-Type-Options: nosniff
X-Frame-Options: DENY

<!DOCTYPE html>
<html lang="en-US">
<head>
  <meta charset="utf-8" >
  <title>template with dictionary</title>
</head>
<body>
  <h1>CSC 108 information</h1>
  <p><strong> Course Name:</strong>Web programming 1</p>
  <p><strong> Course no:</strong> csc108</p>
  <p><strong> Location:</strong> Middletown campus </p>
</body>
</html>

(orname) occc@occc-VirtualBox:csc108/project1/django_project/first_lab$
```

15.4 nginx gunicorn

nginx.conf

```
1 user www-data;
2 worker_processes auto;
3 pid /run/nginx.pid;
4 include /etc/nginx/modules-enabled/*.conf;
5
6 events {
7     worker_connections 4096;
8     # multi_accept on;
9 }
10
11 http {
12
13     ##
14     # Basic Settings
15     ##
16
17     sendfile on;
18     tcp_nopush on;
19     types_hash_max_size 2048;
20     # server_tokens off;
21
22     # server_names_hash_bucket_size 64;
23     # server_name_in_redirect off;
24
25     include /etc/nginx/mime.types;
26     default_type application/octet-stream;
27
28     ##
29     # SSL Settings
30     ##
31
32     ssl_protocols TLSv1 TLSv1.1 TLSv1.2 TLSv1.3; # Dropping SSLv3, ref: ↴
33         ↴ POODLE
34     ssl_prefer_server_ciphers on;
35
36     ##
37     # Logging Settings
38     ##
39
40     access_log /var/log/nginx/access.log;
41     error_log /var/log/nginx/error.log;
42
43     upstream django_server {
44         # fail_timeout=0 means we always retry an upstream even if it ↴
45             ↴ failed
46         # to return a good HTTP response
47
48         # for UNIX domain socket setups
49         server unix:/var/run/gunicorn/gunicorn.sock fail_timeout=0;
50
51         # for a TCP configuration
52         # server 192.168.0.7:8000 fail_timeout=0;
53     }
```

```
53 ##  
54 # Gzip Settings  
55 ##  
56  
57 gzip on;  
58  
59 # gzip_vary on;  
60 # gzip_proxied any;  
61 # gzip_comp_level 6;  
62 # gzip_buffers 16 8k;  
63 # gzip_http_version 1.1;  
64 # gzip_types text/plain text/css application/json application/✓  
    ↴ javascript text/xml application/xml application/xml+rss text/✓  
    ↴ javascript;  
65  
66 ##  
67 # Virtual Host Configs  
68 ##  
69  
70 include /etc/nginx/conf.d/*.conf;  
71 include /etc/nginx/sites-enabled/*;  
72 }
```

/etc/nginx/sites-enabled/default

```
1 ##  
2 # You should look at the following URL's in order to grasp a solid ✓  
    ↴ understanding  
3 # of Nginx configuration files in order to fully unleash the power of Nginx✓  
    ↴ .  
4 # https://www.nginx.com/resources/wiki/start/  
5 # https://www.nginx.com/resources/wiki/start/topics/tutorials/✓  
    ↴ config_pitfalls/  
6 # https://wiki.debian.org/Nginx/DirectoryStructure  
7 #  
8 # In most cases, administrators will remove this file from sites-enabled/ ✓  
    ↴ and  
9 # leave it as reference inside of sites-available where it will continue to✓  
    ↴ be  
10 # updated by the nginx packaging team.  
11 #  
12 # This file will automatically load configuration files provided by other  
13 # applications, such as Drupal or Wordpress. These applications will be ✓  
    ↴ made  
14 # available underneath a path with that package name, such as /drupal8.  
15 #  
16 # Please see /usr/share/doc/nginx-doc/examples/ for more detailed examples.  
17 ##  
18  
19 # Default server configuration  
20 #  
21 server {  
22     listen 80;  
23     listen [::]:80;
```

```
25
26     # SSL configuration
27     #
28     # listen 443 ssl default_server;
29     # listen [::]:443 ssl default_server;
30     #
31     # Note: You should disable gzip for SSL traffic.
32     # See: https://bugs.debian.org/773332
33     #
34     # Read up on ssl_ciphers to ensure a secure configuration.
35     # See: https://bugs.debian.org/765782
36     #
37     # Self signed certs generated by the ssl-cert package
38     # Don't use them in a production server!
39     #
40     # include snippets/snakeoil.conf;
41
42     root /csc108/project1/django_project/first_lab/static;
43
44     # Add index.php to the list if you are using PHP
45     index index.html index.htm index.php;
46
47     server_name csc108.sunyorange.edu.local;
48
49     location / {
50         # First attempt to serve request as file, then
51         # as directory, then fall back to displaying a 404.
52         proxy_set_header Origin "";
53         try_files $uri @proxy_django_server;
54     }
55
56     location /static {
57         autoindex on;
58         alias /csc108/project1/django_project/first_lab/static;
59     }
60
61     location @proxy_django_server {
62         proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
63         proxy_set_header X-Forwarded-Proto $scheme;
64         proxy_set_header Host $http_host;
65         # we don't want nginx trying to do something clever with
66         # redirects, we set the Host: header above already.
67         proxy_set_header X-Real-IP $remote_addr;
68         proxy_redirect off;
69         proxy_pass http://django_server;
70     }
71
72     # pass PHP scripts to FastCGI server
73     #
74     location ~ \.php$ {
75         proxy_set_header Origin "";
76         include snippets/fastcgi-php.conf;
77
78         # With php-fpm (or other unix sockets):
79         fastcgi_pass unix:/run/php/php8.1-fpm.sock;
80         #   # With php-cgi (or other tcp sockets):
81         #   fastcgi_pass 127.0.0.1:9000;
```

```
82    }
83
84    # deny access to .htaccess files, if Apache's document root
85    # concurs with nginx's one
86    #
87    #location ~ /\.ht {
88        #    deny all;
89        #}
90    }
91
92
93 # Virtual Host configuration for example.com
94 #
95 # You can move that to a different file under sites-available/ and symlink ↵
96 # to sites-enabled/ to enable it.
97 #
98 #server {
99     #    listen 80;
100    #    listen [::]:80;
101    #
102    #    server_name example.com;
103    #
104    #    root /var/www/example.com;
105    #    index index.html;
106    #
107    #    location / {
108        #        try_files $uri $uri/ =404;
109        #    }
110    #}
```

Test nginx config files and reload nginx.

```
(orange) occc@occc-VirtualBox:/csc108/project1/django_project/first_lab$ \
> http GET http://csc108.sunyorange.edu.local/csc108/list_example/
HTTP/1.1 200 OK
Connection: keep-alive
Content-Encoding: gzip
Content-Type: text/html; charset=utf-8
Cross-Origin-Opener-Policy: same-origin
Date: Thu, 15 Jun 2023 15:50:45 GMT
Referrer-Policy: same-origin
Server: nginx/1.18.0 (Ubuntu)
Transfer-Encoding: chunked
X-Content-Type-Options: nosniff
X-Frame-Options: DENY
  

<!DOCTYPE html>
<html lang="en-US">
  <head>
    <meta charset="utf-8" >
    <title>template with dictionary</title>
  </head>
  <body>
    <h1>CSC 108 information</h1>
    <p><strong> Course Name:</strong>Web programming 1</p>
    <p><strong> Course no:</strong> csc108</p>
    <p><strong> Location:</strong> Middletown campus </p>
  </body>
</html>

(orange) occc@occc-VirtualBox:/csc108/project1/django_project/first_lab$
```

CSC 108 web dev 1

The screenshot shows a browser's developer tools open. The DOM tree on the left displays the HTML structure of a page, including comments like 'OU Search Ignore Start Here' and 'Google Tag Manager'. The styles panel on the right shows CSS rules for the 'body' element, including 'border-top: 1px solid black' and 'font-family: sans-serif; -webkit-font-smoothing: antialiased;'. The console tab at the bottom has several error messages related to failed resource loads and source maps.

debug toolbar

<https://django-debug-toolbar.readthedocs.io/en/latest/index.html>

NOT TO BE INSTALLED ON PRODUCTION ENVIRONMENT!!!!!!!!!!!!!!

16.1 Installation

```
python -m pip install django-debug-toolbar
```

```
(orange) occc@occc-VirtualBox:/csc108/project1/mywebsite$ python -m pip install django-debug-toolbar
Collecting django-debug-toolbar
  Downloading django_debug_toolbar-4.1.0-py3-none-any.whl (222 kB)
    222.5/222.5 KB 3.8 MB/s eta 0:00:00
Requirement already satisfied: sqlparse>=0.2 in /csc108/project1/orange/lib/python3.10/site-packages (from django-debug-toolbar) (0.4.4)
Requirement already satisfied: django>=3.2.4 in /csc108/project1/orange/lib/python3.10/site-packages (from django-debug-toolbar) (4.2.2)
Requirement already satisfied: asgiref<4,>=3.6.0 in /csc108/project1/orange/lib/python3.10/site-packages (from django>=3.2.4->django-debug-toolbar) (3.7.2)
Requirement already satisfied: typing-extensions>=4 in /csc108/project1/orange/lib/python3.10/site-packages (from asgiref<4,>=3.6.0->django>=3.2.4->django-debug-toolbar) (4.6.3)
Installing collected packages: django-debug-toolbar
Successfully installed django-debug-toolbar-4.1.0
(orange) occc@occc-VirtualBox:/csc108/project1/mywebsite$
```

16.1.1 pre requisites

In your settings.py file make sure you have in the installed apps:
`django.contrib.staticfiles`

Register app

```
INSTALLED_APPS = [  
    'django.contrib.admin',  
    'django.contrib.auth',  
    'django.contrib.contenttypes',  
    'django.contrib.sessions',  
    'django.contrib.messages',  
    'django.contrib.staticfiles',  
    'mainpage',  
    'users',  
    'faculty',  
    'registrar',  
    'students',  
    'crispy_forms',  
    'crispy_bootstrap5',  
]
```

Also make sure you have STATIC_URL enabled.

```
STATIC_URL = "static/"
```

16.1.2 Install the App

debug_toolbar setting.py TEMPLATES

```
1 TEMPLATES = [  
2     {  
3         "BACKEND": "django.template.backends.django.DjangoTemplates",  
4         "APP_DIRS": True,  
5         # ...  
6     }  
7 ]
```

debug_toolbar setting.py INSTALLED_APPS

```
1 INSTALLED_APPS = [  
2     # ...  
3     "debug_toolbar",
```

```
4     # ...
5 ]
```

Register app debug toolbar

```
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'mainpage',
    'users',
    'faculty',
    'registrar',
    'students',
    'crispy_forms',
    'crispy_bootstrap5',
    'debug_toolbar',
]
```

16.1.3 Add the URLs

debug toolbar main urls.py

```
1 from django.contrib import admin
2 from django.urls import include, path
3 #new
4 from django.contrib.auth import views as auth_views
5 #debug toolbar
6 import debug_toolbar
7 from django.conf import settings
8
9
10 urlpatterns = [
11     path("csc108/", include("csc108.urls")),
12     path('admin/', admin.site.urls),
13     path("", include('users.urls')),
14     path("", include('csc108.urls')),
```

```
15     path("", include('faculty.urls')),
16     path("", include('registrar.urls')),
17     path("", include('students.urls')),
18     path("__debug__/", include("debug_toolbar.urls")),
19 ]
```

16.1.4 Add the Middleware

settings.py MIDDLEWARE

```
1 MIDDLEWARE = [
2     # ...
3     "debug_toolbar.middleware.DebugToolbarMiddleware",
4     # ...
5 ]
```

16.1.5 Configure Internal IPs

settings.py INTERNAL_IPS

```
1 INTERNAL_IPS = [
2     "127.0.0.1",
3 ]
```

The screenshot shows a Firefox browser window with two tabs: "College main web page" and "Logged out | Django site admin". The address bar displays "csc108.sunyorange.edu.local:8000". The main content area shows a website for SUNY Orange. The header includes a logo, "Home", "About", and "Log out". A large orange button in the center contains the text "#SOconvenient". Below it, a blue background with white curved lines displays the text "Our campuses are conveniently located in Middletown and Newburgh.". The footer is dark with white text, divided into sections: "SUNY ORANGE", "ACADEMICS", "USEFUL LINKS", and "CONTACT". Under "SUNY ORANGE", there are links for "Middletown campus:" and "Newburgh campus:". Under "ACADEMICS", there are links for "Registrar", "Degrees", "Financial Aid", "Your Account", "News", and "Help". The "CONTACT" section lists "Cache", "Signals", "Middletown Campus", "115 South Street", "Middletown, NY 10540", and "Profiling". On the right side of the browser, a developer toolbar is open, showing performance metrics: History (Django 4.2.2), Versions (Django 4.2.2), Time (CPU: 38.07ms (75.51ms)), Settings, Headers, Request (home), SQL (2 queries in 5.23ms), Static files (5 files used), Templates (csc108/home.html), Cache (0 calls in 0.00ms), Signals (36 receivers of 15 signals), Middletown Campus (Intercept redirects), 115 South Street (Intercept redirects), Middletown, NY 10540 (Intercept redirects), and Profiling.

© 2023 Copyright: [CSC 108 Web Programming 1](#)



Django-extensions

17.1 ➤ Install django extensions

<https://django-extensions.readthedocs.io/en/latest/index.html>

```
(orange) occc@occc-VirtualBox:/csc108/project1/mywebsite$ pip install django-extensions
```

17.1.1 configuration

You will need to add the django_extensions application to the INSTALLED_APPS setting of your Django project settings.py file.

settings add to apps

```
1 INSTALLED_APPS = [  
2     ...  
3     'csc108',  
4     'users',  
5     'faculty',  
6     'registrar',  
7     'students',  
8     "crispy_forms",  
9     "crispy_bootstrap5",  
10    "debug_toolbar",  
11    'django_extensions', #add this to enable django-extensions  
12 ]
```

17.2 generate ERD diagrams

17.2.1 Install pre req

```
sudo apt install python3-pygraphviz
```

```
(orange) ...mywebsite$ python manage.py graph_models -a -g -o imagefile_name.png
```

