# Setup

1. Make sure you have an Integrated Development Environment for Python, such as:
   1. PyCharm
      1. <https://www.jetbrains.com/pycharm/download/>
   2. Visual Studio Code (We’ve been using this)
      1. <https://code.visualstudio.com/download>
      2. Web browser version <https://vscode.dev/>
2. Make sure you have Python installed (We’ve been using 3.10.7)
   1. <https://www.python.org/downloads/>
3. Join the GitHub team – then make your own fork of “BackendDev-repo” repository
   1. If you are from another team **and don’t plan on pushing any changes,** you can just clone [the original repo](https://github.com/cs261f22/BackendDev-repo).
4. cd to the location you want the repo folder to be and clone your fork (or the original repo) using git clone <URL>
   1. You should now have a folder named “BackendDev-Repo”
5. Once within your project folder in your terminal, create a python virtual environment to install all the specific libraries we want for backend
   1. python3 –m venv .venv
6. Activate the virtual environment (assuming it’s named .venv)
   1. for Unix (macOS, linux..)
      1. .venv/bin/activate
   2. for Windows
      1. In CMD: .venv/Scripts/activate.bat
      2. In PowerShell: .venv/Scripts/Activate.ps1
         1. If you get a “cannot be loaded because running scripts is disabled on this system.” error, run the following command and type “Y” when prompted:

Set-ExecutionPolicy RemoteSigned -Scope CurrentUser

1. Install needed libraries to your virtual environment (using the requirements.txt file)
   1. Make sure you’re active inside your virtual environment. You can do this by verifying that it says (.venv) to the left of current location in your terminal
   2. Run pip install -r BackEndDev-Repo/requirements.txt
   3. Check what is installed within your virtual environment using the pip freeze command (this should be same output as the content of the requirements.txt file)
2. To get out of the virtual environment, you can type deactivate. Note that the virtual environment will need to be activated in order to run the API server.

# Startup Script

We have created a startup script that will configure needed permissions and users and add dummy data to the database.

Before running it, we must create the database. In order to do that, while in your virtual environment (see step 6 above for how to activate it), run   
python manage.py makemigrations data. This stages changes to the databases.   
**Then, run python manage.py migrate.** This actually makes those changes.  
This should create a file called db.sqlite3 in the databases folder. This is an empty database **with no data**. In order to fill it with data, you should run the startup script.

Once you have created the database, you can run the startup script by calling the following command:  
python manage.py db\_startup

Once the script has run, you should have a database file called db.sqlite3 in the databases folder with dummy data. In addition, there are some users that are automatically created, including a superuser. The pre-configured credentials for the superuser are:  
username: superuser  
password: xu261backend\_su

If you would like to create more users, make sure you are logged in as a user with the appropriate permissions, as described in the Permissions section of [the API Documentation](https://myxavier.sharepoint.com/sites/CS261Fall20222/Shared%20Documents/Web%20Backend/API%20Documentation.docx), and call the register endpoint on the type of user you want to create, as described in the API Endpoints section of the API Documentation.

# Running the Server

In order make API calls, the Django server must also be running locally.

After you’ve run the startup script, and while in your virtual environment (see step 6 in [the Setup section](#_Setup) for how to activate it), run python manage.py runserver to run the server. You can press CTRL + C to stop the server. The server is located at <http://127.0.0.1:8000/>. If you would like to quickly verify that it is up and running, you can go to that link in your web browser.

If the server is started successfully, you should see a message similar to the one below in the terminal:  
System check identified no issues (0 silenced).  
November 16, 2022 - 15:10:31  
Django version 4.1.1, using settings 'BackendDev.settings'  
Starting development server at http://127.0.0.1:8000/  
Quit the server with CTRL-BREAK.

# Tests

If you would like to confirm that the API is working as expected, you can run our tests. These are located in data/tests/ if you would like to take a look at the code for them. Here is how to get your tests working:

1. Ensure that the server is not running and there is no testing database in the databases folder. If there is, run rm databases/testing.sqlite3. Additionally, make sure that your virtual environment is activated.
2. Create a clean version of the testing database by running python manage.py migrate-and-setup --testing.
3. Run the server using the testing database by running python manage.py runserver   
   --settings=BackendDev.testSettings.
   1. If you are using the testing GUI that is integrated into an IDE such as VSCode, you can do this in the same terminal window that you’ve been doing the rest in.
   2. If you are planning on testing using the command line, open a new terminal window.
      1. You can do this in Powershell by running start powershell.
4. Run the tests (**while the server is still running**).
   1. If you are using the integrated testing GUI in VSCode, [here](https://code.visualstudio.com/docs/python/testing) is some of their documentation on Python testing.
   2. If you are using the command line, you can run pytest data/tests (assuming you are in the base directory). This will run all tests in the data/tests directory, so if you want to run a specific file, you can use the file path in place of the directory.
   3. Running the tests can take upwards of 45 seconds to a minute and a half depending on the power of your machine.
5. Once the tests are done running, it is recommended that you stop the server and then delete the testing database using rm databases/testing.sqlite3.

We are using the Pytest library for our tests. You can see documentation for that [here](https://docs.pytest.org/en/7.1.x/contents.html).   
For Django specific tests, we’re using the Pytest-Django library. You can see documentation for that [here](https://pytest-django.readthedocs.io/en/latest/).