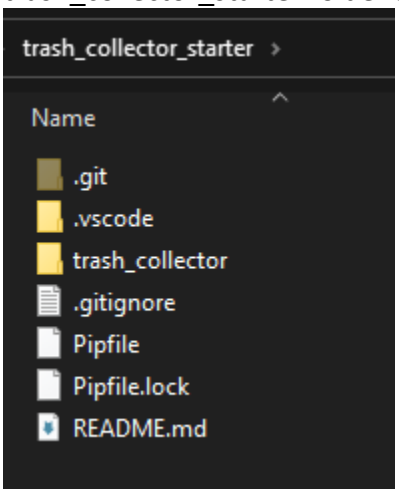
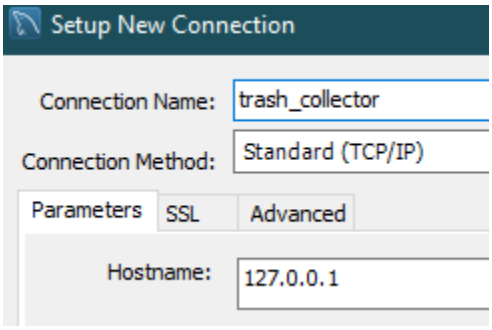
Trash Collector Collaboration Setup steps

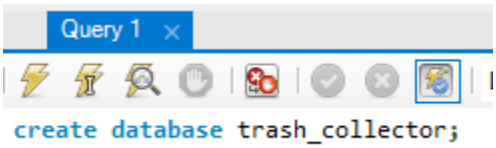
1. All group members unzip the trash_collector_starter.zip file. This will create a folder named 'trash_collector_starter'. Move this folder to the desired location where you keep your projects.
2. Decide who will create the repo.
3. One person creates a github repo named '**Django_Trash_Collector**'. Create the repo with a README. No need for a gitignore, one is included in the starter.
4. Repo creator invites others as collaborators, collaborators check email and accept invite.
5. All group members clone down the repo. This will create a folder called 'Django_Trash_Collector' that will contain a README and a hidden .git folder.
6. Copy the README and hidden .git folder from 'Django_Trash_Collector' folder and Paste into the 'trash_collector_starter' folder (folder with Pipfile & gitignore). trash_collector_starter folder should now contain the following items:



7. Everyone deletes the now empty 'Django_Trash_Collector' folder.
8. In MySQL workbench, all partners create a new database connection called 'trash_collector'



9. In MySQL workbench, all partners run the following SQL query:



10. All partners open **trash_collector_starter** project folder in VS Code

IMPORTANT: Always open the trash_collector_starter folder as the project in Vs Code! Do not open the nested folder or an outer containing folder.

11. In VSCode, all partners open a new terminal and enter:
    pipenv install
    pipenv shell
12. All Partners go to View > Command Palette, Type in 'Python Select' to bring up the Python: Select Interpreter option and select the python.exe from the .virutalenvs\trash_collector_starter folder that was just created.
13. All partners change the DATABASES object in local_settings.py to reflect personal MySQL Db password.
14. **python manage.py migrate**
15. **python manage.py createsuperuser**
16. Start program, go to 127.0.0.1:8000/admin
17. Log into admin site as superuser account
18. In admin site, create groups "Customers" and "Employees" (case sensitive)
19. Ready to go!

First Steps when diving into project:

Discuss Django MVT pattern with partner!

Examine important files in customers app: models.py, urls.py, views.py, existing templates. **Read comments & look for TODOS!**

Add properties to customer model

Add a 'create' view function & template for customers app

General Tips:

'accounts' app is complete & will not need to be modified.

Beware Premature Optimization, always be chasing MVP!

Separate 'wants' from 'needs'. If a user story doesn't ask for it, do you even need to do it?

Get it working, then make it better!

For migrations:

Migration files include a time stamp, so even two partners who migrate the same exact same model properties will get a merge conflict from the timestamp differing.

Have one person in charge of migrations. They change the model, enter the command ‘python manage.py makemigrations’, test it is good by running ‘python manage.py migrate’. If the migration works, they make a new commit to push the created migration file to the repo.

Collaborators will then pull down the migration next time they make a commit.

If you nuke your DB:

Re-migrate (python manage.py migrate)

Re-create admin superuser

Re-create Customer & Employee groups in Admin interface

If you re-clone the repo:
local_settings.py will be gone, be sure to save the one from previous version, or have partner send it over slack if you lost it.