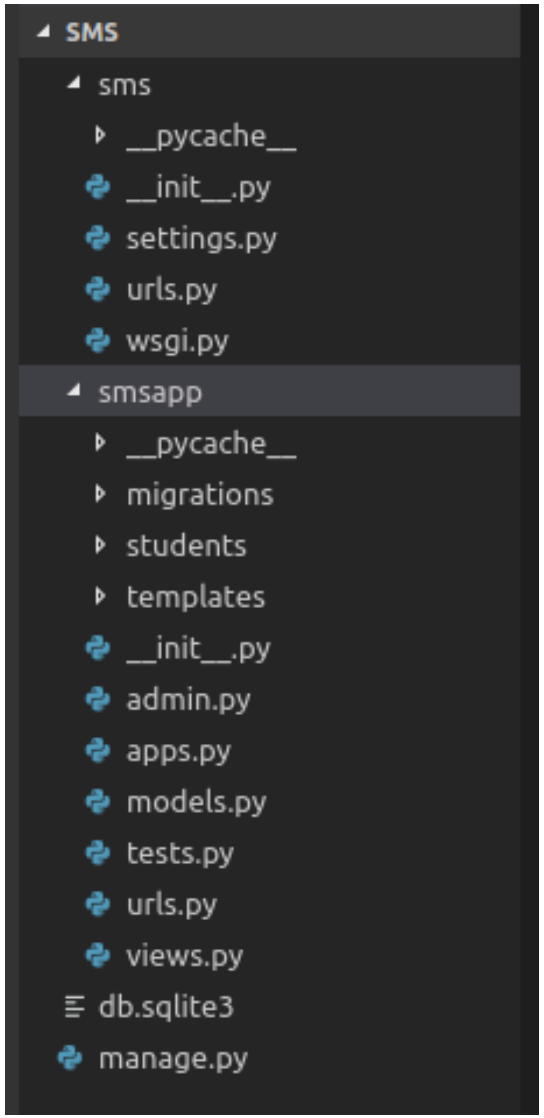
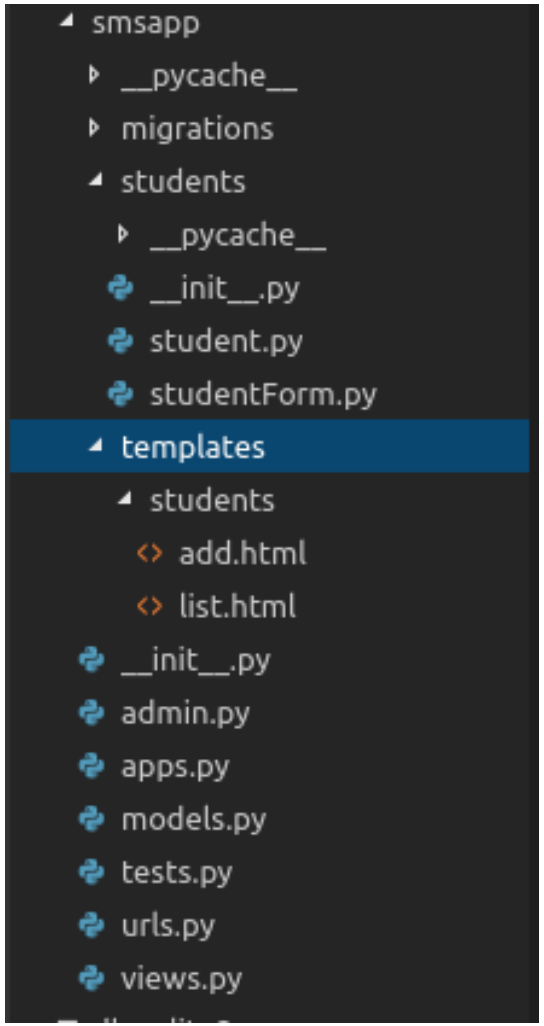
Python
Server-Side Framework
Files & Databases

Lab 8

Review

- We created a project called “sms” and created an application within it called “smsapp”
- Within the smsapp, we specified:
  - Models (Students & Courses)
  - Model Forms (StudentForm and CourseForm)
  - Views
  - Templates
  - Routes





Review

- Importantly, while we are storing the data to a SQLite Database, this database cannot be used for production systems.
- To develop more realistic web application, we will utilize a MySQL relational database management system (RDBMS)

Install the dependencies

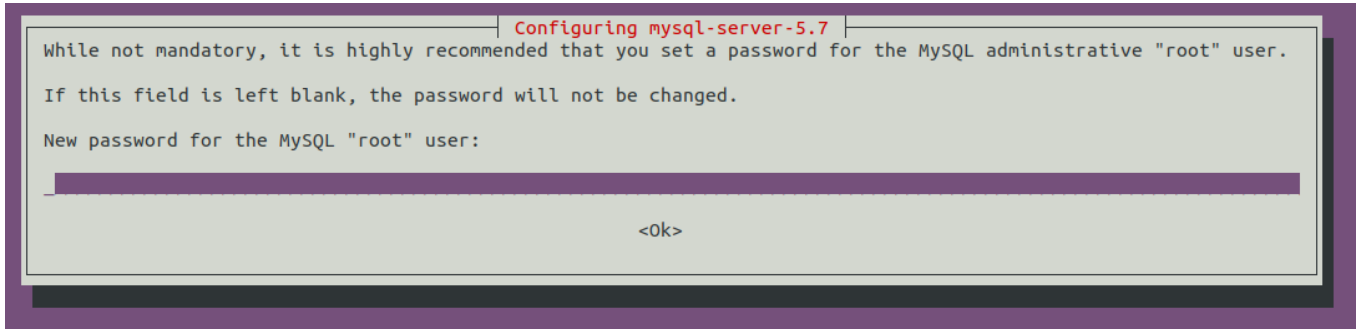
- Start the virtual environment

```
adminuser@AdminVirt:~$ cd dev/python-sms/
adminuser@AdminVirt:~/dev/python-sms$ source venv/bin/activate
(venv) adminuser@AdminVirt:~/dev/python-sms$
```

- Install mysql database
  - sudo apt-get install mysql-server mysql-client –y

```
(venv) adminuser@AdminVirt:~/dev/python-sms$ sudo apt-get install mysql-server mysql-client -y
[sudo] password for adminuser:
Reading package lists... Done
Building dependency tree
Reading state information... Done
```

- Add enter the credentials to access the DBMS
  - I would suggest to use “adminuser”



Install the dependencies

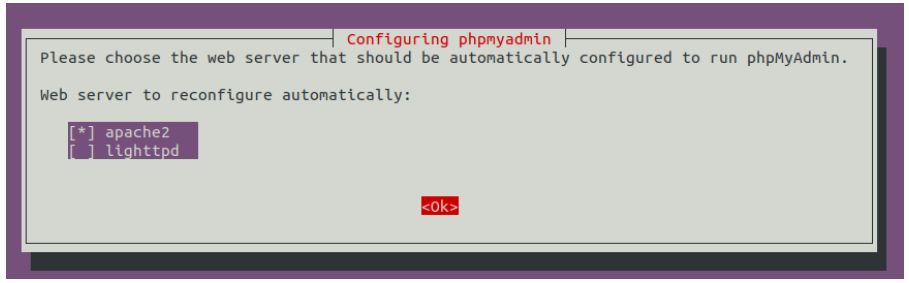
- Test connecting to the database:
  - mysql –u root -p

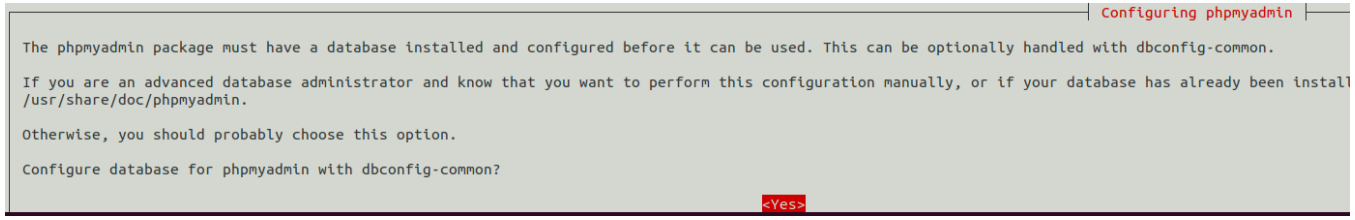
```
(venv) adminuser@AdminVirt:~/dev/python-sms$ mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 4
Server version: 5.7.21-0ubuntu0.17.10.1 (Ubuntu)

Copyright (c) 2000, 2018, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
```

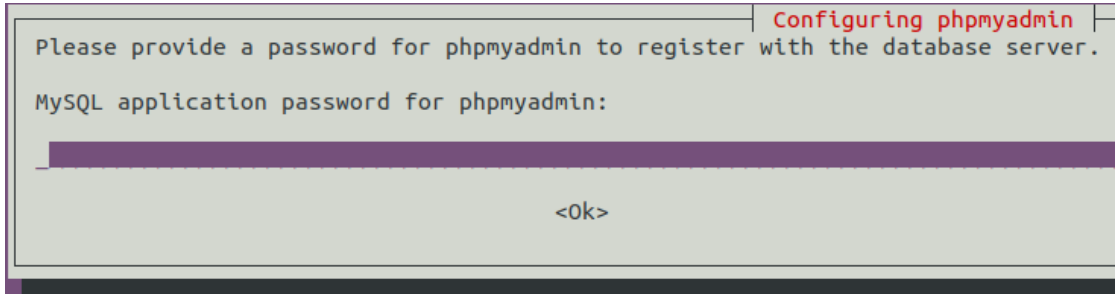
Install the dependencies

- Install the PHPMyAdmin program to manage the MySQL database visually
  - sudo apt-get install phpmyadmin -y

```
(venv) adminuser@AdminVirt:~/dev/python-sms$ sudo apt-get install phpmyadmin -y
Reading package lists... Done
Building dependency tree
Reading state information... Done
```



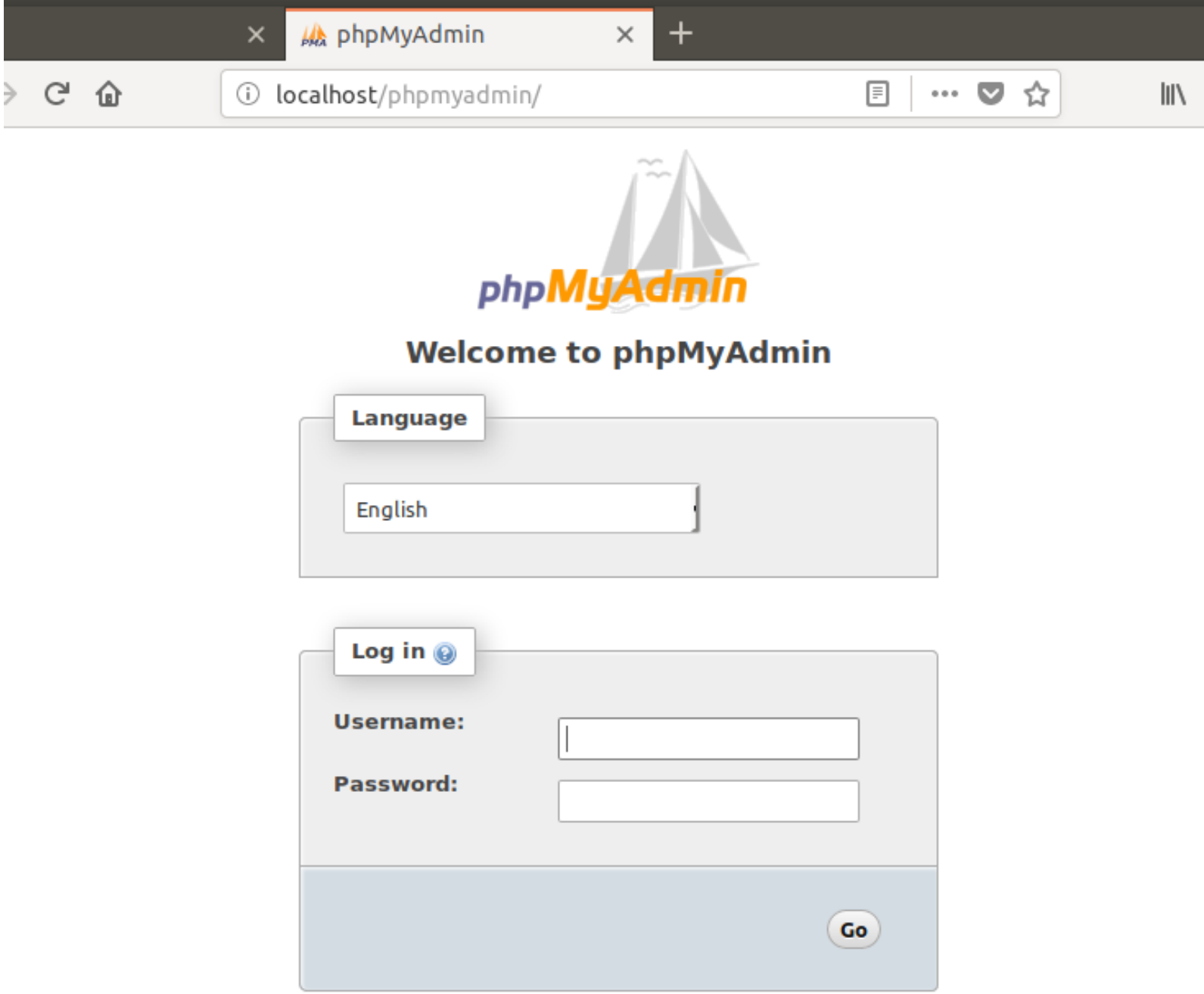




Use the same credentials for the database created previously: adminuser

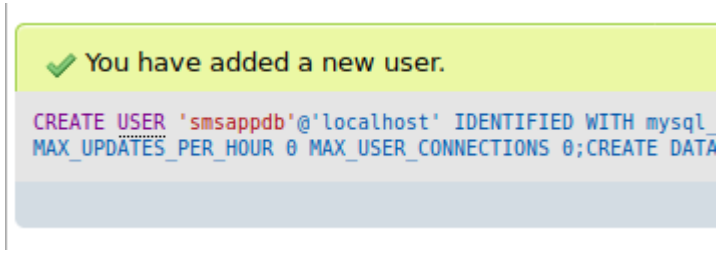
Install the dependencies

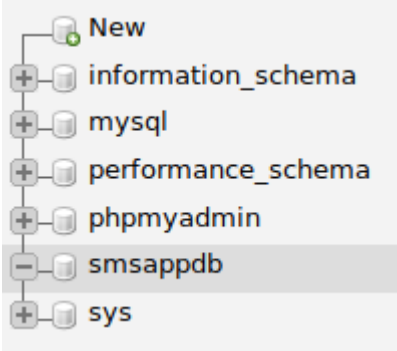
- Test the phpmyadmin by entering the url: http://localhost/phpmyadmin/
- Enter the credentials:
  - Username: root
  - Password: adminuser

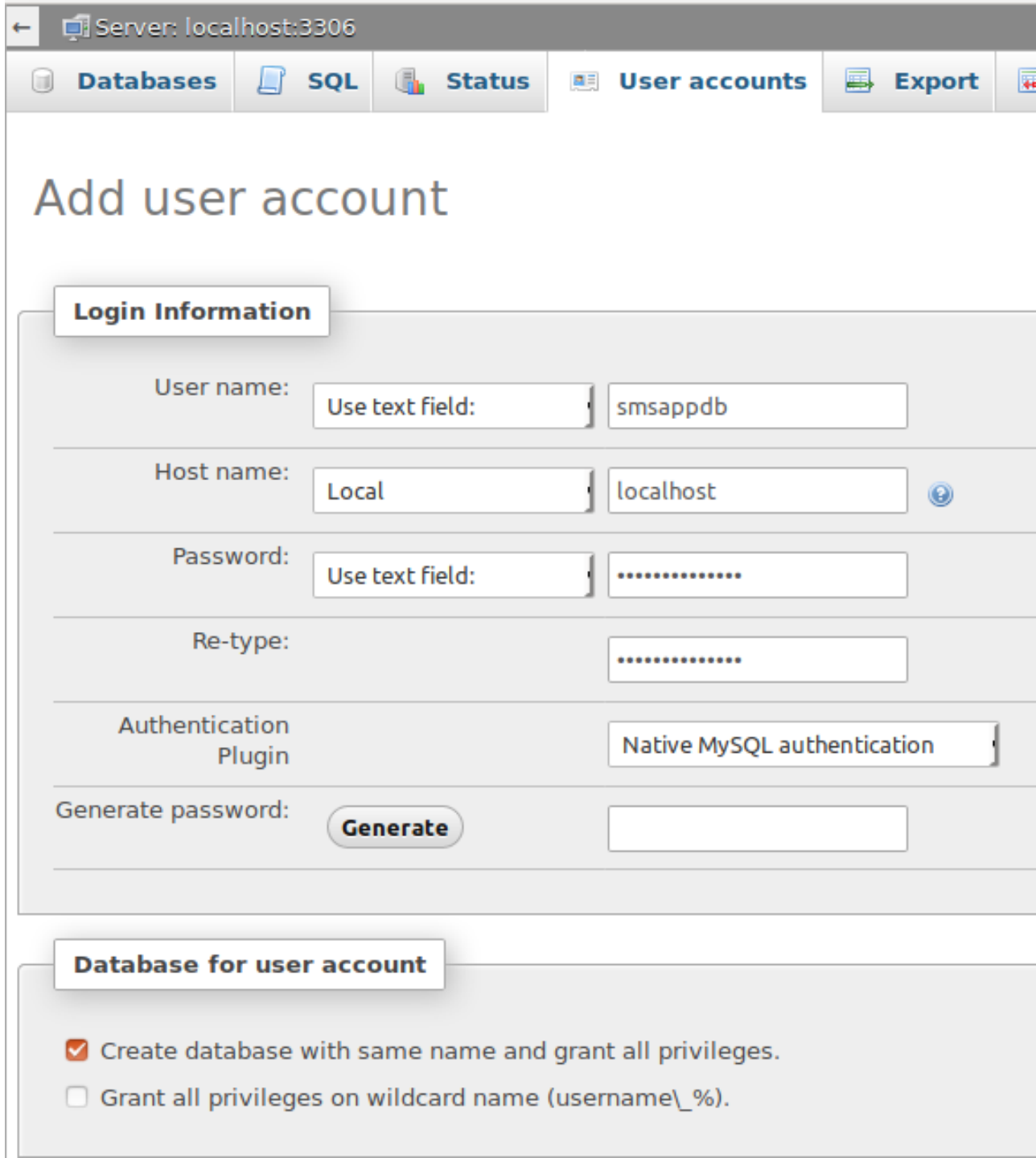


Creating User and Database

- We will create a database user:
  - Username: smsappdb
  - Hostname: localhost
  - Password: smsP@ssword1^2
- Ensure to select “create database with same name and grant all privileges”







Install the dependencies

- Now that we installed the database related software we need to connect our Django application to the database.
- We need to ensure that we have the drivers that allow python (and therefore Django) to communicate with the browser:
  - sudo apt-get install python3-dev libmysqlclient-dev
- Ensuring that we are within the virtual environment for our app and install the mysql client

```
(venv) adminuser@AdminVirt:~/dev/python-sms$ sudo apt-get install python3-dev libmysqlclient-dev -y
Reading package lists... Done
Building dependency tree
Reading state information... Done
python3-dev is already the newest version (3.6.3-0ubuntu2).
python3-dev set to manually installed.
```

```
(venv) adminuser@AdminVirt:~/dev/python-sms$ pip install mysqlclient
Collecting mysqlclient
  Downloading mysqlclient-1.3.12.tar.gz (89kB)
    100% |                                | 92kB 671kB/s
Building wheels for collected packages: mysqlclient
```

```
Installing collected packages: mysqlclient
  Running setup.py install for mysqlclient ... done
Successfully installed mysqlclient-1.3.12
(venv) adminuser@AdminVirt:~/dev/python-sms$
```

Add the MySQL database connection to the application

EXPLORER

OPEN EDITORS
- settings.py sms

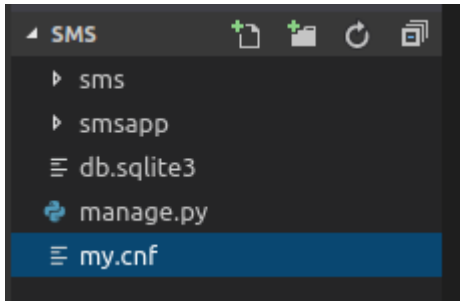
SMS
- sms
  - __pycache__
  - __init__.py
  - settings.py
  - urls.py
  - wsgi.py
- smsapp
  - __pycache__
  - migrations
  - students
  - templates
  - __init__.py
  - admin.py
  - apps.py
  - models.py
  - tests.py
  - urls.py
  - views.py
- db.sqlite3
- manage.py

settings.py

```python
                'django.contrib.auth.context_processors.auth',
                'django.contrib.messages.context_processors.messages',
            ],
        },
    },
]

WSGI_APPLICATION = 'sms.wsgi.application'


# Database
# https://docs.djangoproject.com/en/2.0/ref/settings/#databases

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': os.path.join(BASE_DIR, 'db.sqlite3'),
    }
}


# Password validation
# https://docs.djangoproject.com/en/2.0/ref/settings/#auth-password-validators

AUTH_PASSWORD_VALIDATORS = [
    {
        'NAME': 'django.contrib.auth.password_validation.UserAttributeSimilarit
    },
```

Add the MySQL database connection to the application

```python
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.mysql',
        'OPTIONS': {
            'read_default_file': os.path.join(BASE_DIR, 'my.cnf'),
        }
    }
}
```



settings.py | my.cnf

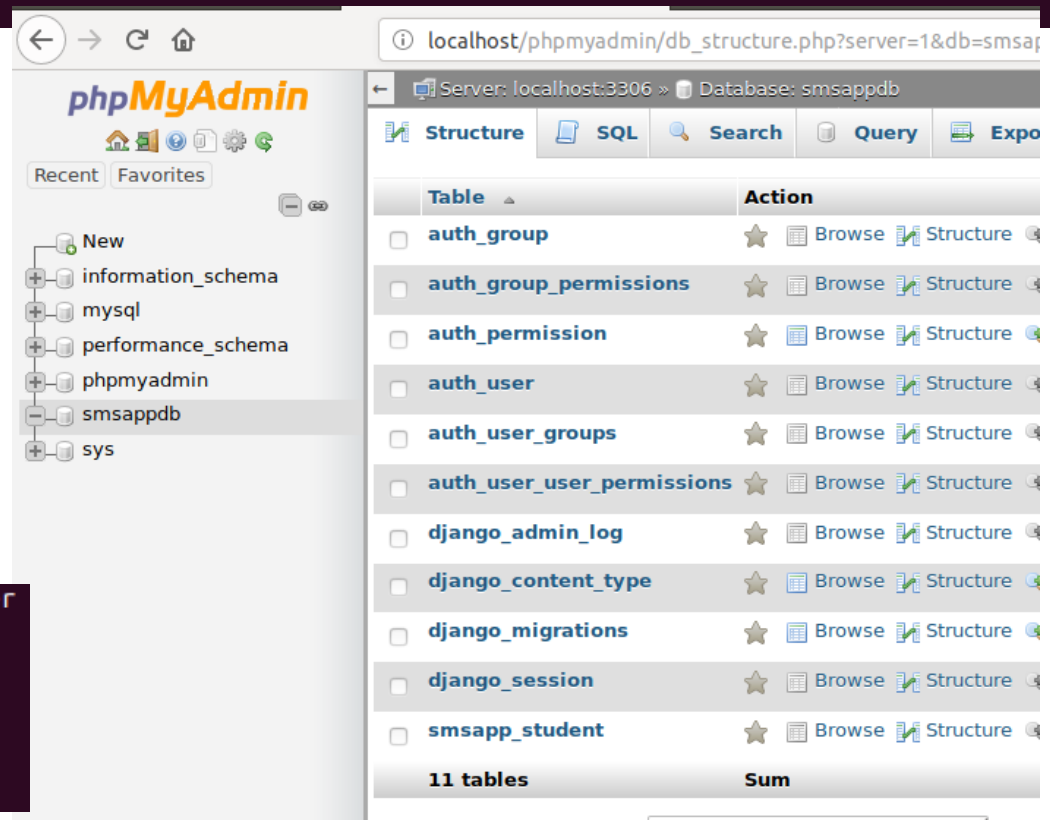
```
[client]
database = smsappdb
user = smsappdb
password = smsP@ssword1^2
default-character-set = utf8
```

- Install the database tables and other related specifications developed:
  - python manage.py migrate

```
(venv) adminuser@AdminVirt:~/dev/python-sms/sms$ python manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, sessions, smsapp
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 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 sessions.0001_initial... OK
  Applying smsapp.0001_initial... OK
```

- Then run server

```
(venv) adminuser@AdminVirt:~/dev/python-sms/sms$ python manage.py runserver
Performing system checks...

System check identified no issues (0 silenced).
March 16, 2018 - 05:32:11
Django version 2.0.3, using settings 'sms.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CONTROL-C.
```



- Recreate superuser
  - python manage.py createsuperuser

```
^C(venv) adminuser@AdminVirt:~/dev/python-sms/sms$ python manage.py createsuperuser
Username (leave blank to use 'adminuser'): adminuser
Email address: test@test.com
Password:
Password (again):
Superuser created successfully.
(venv) adminuser@AdminVirt:~/dev/python-sms/sms$
```

Activity 1

- Recreate Student and Course records using the admin panel and the form created from the lab 7
- After view the records in the PHPMyadmin interface

File Management

Creating a profile Image

- Our general steps includes:
  - Modify Student model to accept the image
  - Install Pillow
  - Update the database using migrations
  - Modify the form
  - Upload image and store path to database

Modify Student model to accept the image

student.py

```python
import uuid
from django.db import models
from django.db.models import (
    UUIDField,
    CharField,
    IntegerField,
    BooleanField,
    DateTimeField,
    ImageField
)
```

SMS

- sms
- smsapp
  - __pycache__
  - migrations
  - students
    - __pycache__
    - __init__.py
    - student.py
    - studentForm.py
  - templates
  - __init__.py
  - admin.py
  - apps.py
  - models.py
  - tests.py
  - urls.py
  - views.py
- db.sqlite3
- manage.py
- my.cnf

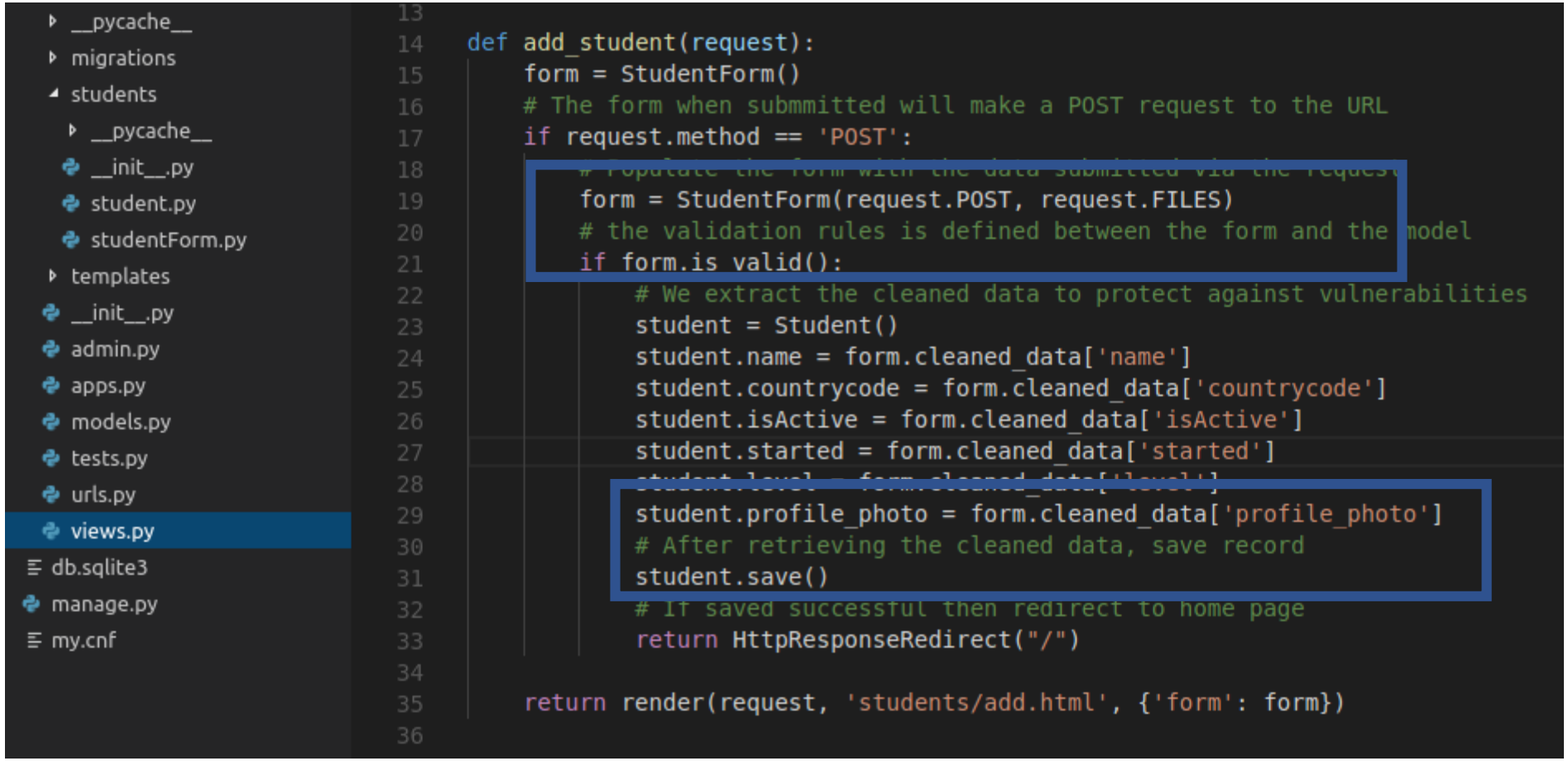
```python


class Student(models.Model):
    id = UUIDField(primary_key=True, default=uuid.uuid4, editable=False)
    name = CharField(max_length=100)
    countrycode = CharField(max_length=3, blank=False, default='TTO')
    isActive = BooleanField(default=False)
    started = IntegerField()
    created = DateTimeField(auto_now_add=True)
    UNDERGRAD = "UG"
    POSTGRAD = "PG"
    LEVEL_CHOICE = (
        (UNDERGRAD, "Undergraduate"),
        (POSTGRAD, "Postgraduate")
    )
    level = CharField(max_length=2, choices=LEVEL_CHOICE, default=UNDERGRAD)
    # Adding the field to accept/store iamges
    profile_photo = ImageField(null=True, upload_to='img/profiles',
    verbose_name="Profile Photo")

    class Meta:
        ordering = ('created',)

```

- Install the Pillow package to be able to process the image
- Required for using the ImageField

```
^C(venv) adminuser@AdminVirt:~/dev/python-sms/sms$ pip install Pillow
Collecting Pillow
  Downloading Pillow-5.0.0-cp36-cp36m-manylinux1_x86_64.whl (5.9MB)
    100% |                                        | 5.9MB 257kB/s
Installing collected packages: Pillow
Successfully installed Pillow-5.0.0
(venv) adminuser@AdminVirt:~/dev/python-sms/sms$
```

- Perform database migration to update the database table with the specifications

```
(venv) adminuser@AdminVirt:~/dev/python-sms/sms$ python manage.py makemigrations
Migrations for 'smsapp':
  smsapp/migrations/0002_student_profile_photo.py
    - Add field profile_photo to student
(venv) adminuser@AdminVirt:~/dev/python-sms/sms$ python manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, sessions, smsapp
Running migrations:
  Applying smsapp.0002_student_profile_photo... OK
(venv) adminuser@AdminVirt:~/dev/python-sms/sms$
```


__pycache__
migrations
students
__pycache__
__init__.py
student.py
studentForm.py
templates
__init__.py
admin.py
apps.py
models.py
tests.py
urls.py
views.py
db.sqlite3
manage.py
my.cnf
13
14 def add_student(request):
15 form = StudentForm()
16 # The form when submmitted will make a POST request to the URL
17 if request.method == 'POST':
18 # Populate the form with the data submitted via the request
19 form = StudentForm(request.POST, request.FILES)
20 # the validation rules is defined between the form and the model
21 if form.is_valid():
22 # We extract the cleaned data to protect against vulnerabilities
23 student = Student()
24 student.name = form.cleaned_data['name']
25 student.countrycode = form.cleaned_data['countrycode']
26 student.isActive = form.cleaned_data['isActive']
27 student.started = form.cleaned_data['started']
28 student.level = form.cleaned_data['level']
29 student.profile_photo = form.cleaned_data['profile_photo']
30 # After retrieving the cleaned data, save record
31 student.save()
32 # If saved successful then redirect to home page
33 return HttpResponseRedirect("/")
34
35 return render(request, 'students/add.html', {'form': form})
36

Server: localhost:3306 » Database: smsappdb » Table: smsapp_student

Browse | Structure | SQL | Search | Insert | Export | Import | Privileges | Operations | Tracking | Triggers

Showing rows 0 - 1 (2 total, Query took 0.0006 seconds.)

```
SELECT * FROM `smsapp_student`
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

Show all | Number of rows: 25 Filter rows: Search this table Sort by key: None

+ Options

←T→	id	name	countrycode	isActive	started	created	level	profile_photo
Edit Copy Delete	2451cbc1fd8a4aee8d40f3ce22243f21	Jane Doe	TTO	1	2018	2018-03-16 07:16:51.840430	UG	img/profiles/Movie-Night.jpg
Edit Copy Delete	82ff93b2d8564297a6138fa17ffc4135	John Doe	TTO	1	2017	2018-03-16 05:38:30.743325	UG	*NULL*

Check all *With selected:* Edit Copy Delete Export

Show all | Number of rows: 25 Filter rows: Search this table Sort by key: None

Query results operations

Print Copy to clipboard Export Display chart Create view

Bookmark this SQL query

Reference

1. https://docs.djangoproject.com/en/2.0/ref/databases/
2. https://www.digitalocean.com/community/tutorials/how-to-create-a-django-app-and-connect-it-to-a-database
3. https://docs.djangoproject.com/en/2.0/topics/http/file-uploads/