Django Todolist

1. Setup the env (with python3 installed)

```
JunchengdeMacBook-Pro-2:THEHACK junchengzhu$ pip3 install django
Collecting django
  Downloading https://files.pythonhosted.org/packages/c7/87/fbd666c4f87591ae25b7bb37
4298e8629816e87193c4099d3608ef11fab9/Django-2.1.7-py3-none-any.whl (7.3MB)
    100% |████████████████████████████████| 7.3MB 4.3MB/s
Collecting pytz (from django)
  Downloading https://files.pythonhosted.org/packages/61/28/1d3920e4d1d50b19bc5d2439
3a7cd85cc7b9a75a490570d5a30c57622d34/pytz-2018.9-py2.py3-none-any.whl (510kB)
    100% |████████████████████████████████| 512kB 17.7MB/s
Installing collected packages: pytz, django
Successfully installed django-2.1.7 pytz-2018.9
```

2. init the project

```
JunchengdeMacBook-Pro-2:THEHACK junchengzhu$ django-admin.py startproject todoapp
JunchengdeMacBook-Pro-2:THEHACK junchengzhu$ cd todoapp/
```

```
JunchengdeMacBook-Pro-2:todoapp junchengzhu$ python3 manage.py startapp todolist
JunchengdeMacBook-Pro-2:todoapp junchengzhu$ tree
.
├── manage.py
├── todoapp
│   ├── __init__.py
│   ├── __pycache__
│   │   ├── __init__.cpython-37.pyc
│   │   └── settings.cpython-37.pyc
│   ├── settings.py
│   ├── urls.py
│   └── wsgi.py
└── todolist
    ├── __init__.py
    ├── admin.py
    ├── apps.py
    ├── migrations
    │   └── __init__.py
    ├── models.py
    ├── tests.py
    └── views.py
```

3. Update the settings.py

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

```
TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [os.path.join(BASE_DIR, "templates")],
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [
                'django.template.context_processors.debug',
                'django.template.context_processors.request',
                'django.contrib.auth.context_processors.auth',
                'django.contrib.messages.context_processors.messages',
            ],
        },
    },
]
```

```
PROJECT_ROOT = os.path.dirname(os.path.abspath(__file__))
STATIC_ROOT = os.path.join(PROJECT_ROOT, 'static')
STATICFILES_STORAGE =
'whitenoise.django.GzipManifestStaticFilesStorage'
```

```
    # add this to MIDDLEWARE & delete the STATICFILES_STORAGE line
    'whitenoise.middleware.WhiteNoiseMiddleware',
```

4. Build the views.py

```
    from django.shortcuts import render

    # Create your views here.
    from django.shortcuts import render,redirect
    from .models import TodoList, Category
    def index(request): #the index view
        todos = TodoList.objects.all() #quering all todos with the
    object manager
```

```
    categories = Category.objects.all() #getting all categories with
object manager
    if request.method == "POST": #checking if the request method is
a POST
        if "taskAdd" in request.POST: #checking if there is a
request to add a todo
            title = request.POST["description"] #title
            date = str(request.POST["date"]) #date
            category = request.POST["category_select"] #category
            content = title + " -- " + date + " " + category
#content
            Todo = TodoList(title=title, content=content,
due_date=date, category=Category.objects.get(name=category))
            Todo.save() #saving the todo
            return redirect("/") #reloading the page
        if "taskDelete" in request.POST: #checking if there is a
request to delete a todo
            checkedlist = request.POST["checkedbox"] #checked todos
to be deleted
            for todo_id in checkedlist:
                todo = TodoList.objects.get(id=int(todo_id))
#getting todo id
                todo.delete() #deleting todo
    return render(request, "index.html", {"todos": todos,
"categories":categories})
```

5. Build the models.py

```
from django.db import models

# Create your models here.
from django.utils import timezone

class Category(models.Model): # The Category table name that
inherits models.Model
    name = models.CharField(max_length=100) #Like a varchar
    class Meta:
        verbose_name = ("Category")
        verbose_name_plural = ("Categories")
    def __str__(self):
        return self.name #name to be shown when called
```

```
    class TodoList(models.Model): #Todolist able name that inherits
    models.Model
        title = models.CharField(max_length=250) # a varchar
        content = models.TextField(blank=True) # a text field
        created = models.DateField(default=timezone.now().strftime("%Y-
    %m-%d")) # a date
        due_date = models.DateField(default=timezone.now().strftime("%Y-
    %m-%d")) # a date
        category = models.ForeignKey(Category,
    default="general",on_delete=models.DO_NOTHING) # a foreignkey
        class Meta:
            ordering = ["-created"] #ordering by the created field
        def __str__(self):
            return self.title #name to be shown when called
```

```
JunchengdeMacBook-Pro-2:todoapp junchengzhu$ python3 manage.py makemigrations
Migrations for 'todolist':
  todolist/migrations/0001_initial.py
    - Create model Category
    - Create model TodoList
```

```
JunchengdeMacBook-Pro-2:todoapp junchengzhu$ python3 manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, sessions, todolist
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 sessions.0001_initial... OK
  Applying todolist.0001_initial... OK
```

6. superuser account generate

```
manage.py createsuperuser
```

```
Username (leave blank to use 'dell'): todoapp
Email address: admin@todoapp.com
Password:
Password (again):
Superuser created successfully.
```

So enter a new username, email and password for django admin. Once you've done that, you'll need to start the server using the default django local web server:

```
manage.py runserver 8100
```

7. Templates folders

```
cd todolist
mkdir templates
touch index.html
mkdir css
cd css
touch style.css
```

```
<!DOCTYPE html>
<html >
<head>
  <meta charset="UTF-8">
  <title>TodoApp - Create A Todo With Django</title>
  {% load static %}
  <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0-
beta/css/bootstrap.min.css" integrity="sha384-
/Y6pD6FV/Vv2HJnA6t+vslU6fwYXjCFtcEpHbNJ0lyAFsXTsjBbfaDjzALeQsN6M"
crossorigin="anonymous">
  <link rel="stylesheet" type="text/css"
href="https://maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/font-
awesome.min.css">
```

```
    <link rel="stylesheet" type="text/css" href="{% static
'css/style.css' %}">
</head>
<body>
<div django-app="TaskManager">
    <div class="container">
            <div class="content">
            <h1>TodoApp</h1>
            <p class="tagline">a Django todo app</p>
            <form action="" method="post">
            {% csrf_token %} <!-- csrf token for basic security -->
                <div class="inputContainer">
                    <input type="text" id="description"
class="taskName" placeholder="What do you need to do?"
name="description" required>
                    <label for="description">Description</label>
                </div>
                <div class="inputContainer half last">
                    <i class="fa fa-caret-down selectArrow"></i>
                    <select id="category" class="taskCategory"
name="category_select">
                    <option class="disabled" value="">Choose a
category</option>
                    {% for category in categories %}
                        <option class="" value="{{ category.name }}"
name="{{ category.name }}">{{ category.name }}</option>
                    {% endfor %}
                    </select>
                    <label for="category">Category</label>
                </div>
                <div class="inputContainer half last right">
                    <input type="date" id="dueDate" class="taskDate"
name="date">
                    <label for="dueDate">Due Date</label>
                </div>
                <div class="row">
                    <button class="taskAdd" name="taskAdd"
type="submit"><i class="fa fa-plus icon"></i>Add task</button>
                    <button class="taskDelete" name="taskDelete"
formnovalidate="" type="submit"
onclick="$('input#sublist').click();"><i class="fa fa-trash-o icon">
</i>Delete Tasks</button>
                </div>
            <ul class="taskList">
```

```
            {% for todo in todos %} <!-- django template lang - for
loop -->
                <li class="taskItem">
                    <input type="checkbox" class="taskCheckbox"
name="checkedbox" id="{{ todo.id }}" value="{{ todo.id }}">
                    <label for="{{ todo.id }}"><span
class="complete-">{{ todo.title }}</span></label>
                    <span class="category-{{ todo.category }}">{{
todo.category }}</span>
                    <strong class="taskDate"><i class="fa fa-
calendar"></i>{{ todo.created }} - {{ todo.due_date }}</strong>
                </li>
            {% endfor %}
            </ul><!-- taskList -->
            </form>
            </div><!-- content -->
 </div><!-- container -->
</div>
<script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.1/jquery.min.j
s"></script>
</body>
</html>
```

```
/* basic reset */
*, *:before, *:after {
    -webkit-box-sizing: border-box;
       -moz-box-sizing: border-box;
            box-sizing: border-box;
}
/* app */
html {
    font-size: 100%;
}
body {
    background: #b1f6cb;
    font-family: 'Open Sans',sans-serif;
}
/* super basic grid structure */
.container {
    width: 600px;
    margin: 0 auto;
    background: #ffffff;
    padding: 20px 0;
```

```
    -webkit-box-shadow: 0 0 2px rgba(0,0,0,0.2);
            box-shadow: 0 0 2px rgba(0,0,0,0.2);
}
.row {
    display: block;
    padding: 10px;
    text-align: center;
    width: 100%;
    clear: both;
    overflow: hidden;
}
.half {
    width: 50%;
    float: left;
}
.content {
    background: #fff;
}
/* logo */
h1 {
    font-family: 'Rokkitt', sans-serif;
    color: #666;
    text-align: center;
    font-weight: 400;
    margin: 0;
}
.tagline {
    margin-top: -10px;
    text-align: center;
    padding: 5px 20px;
    font-size: 11px;
    font-weight: 600;
    text-transform: uppercase;
    color: #777;
}
/* inputs */
.inputContainer {
    height: 60px;
    border-top: 1px solid #e5e5e5;
    position: relative;
    overflow: hidden;
}
.inputContainer.last {
    border-bottom: 1px solid #e5e5e5;
    margin-bottom: 20px;
```

```
}
.inputContainer.half.last.right {
    border-left: 1px solid #efefef;
}
input[type="date"], input[type="text"], select {
    height: 100%;
    width: 100%;
    padding: 0 20px;
    position: absolute;
    top: 0;
    vertical-align: middle;
    display: inline-block;
    border: none;
    border-radius: none;
    font-size: 13px;
    color: #777;
    margin: 0;
    font-family: 'Open Sans', sans-serif;
    font-weight: 600;
    letter-spacing: 0.5px;
    -webkit-transition: background 0.3s;
            transition: background 0.3s;
}
input[type="date"] {
    cursor: pointer;
}
input[type="date"]:focus, input[type="text"]:focus, select:focus {
    outline: none;
    background: #ecf0f1;
}
::-webkit-input-placeholder {
    color: lightgrey;
    font-weight: normal;
    -webkit-transition: all 0.3s;
            transition: all 0.3s;
}
::-moz-placeholder {
    color: lightgrey;
    font-weight: normal;
    transition: all 0.3s;
}
::-ms-input-placeholder {
    color: lightgrey;
    font-weight: normal;
    transition: all 0.3s;
```

```
    }
    input:-moz-placeholder {
        color: lightgrey;
        font-weight: normal;
        transition: all 0.3s;
    }
    input:focus::-webkit-input-placeholder {
        color: #95a5a6;
        font-weight: bold;
    }
    input:focus::-moz-input-placeholder {
        color: #95a5a6;
        font-weight: bold;
    }
    .inputContainer label {
        padding: 5px 20px;
        font-size: 11px;
        font-weight: 600;
        text-transform: uppercase;
        color: #777;
        display: block;
        position: absolute;
    }
    button {
        font-family: 'Open Sans', sans-serif;
        background: transparent;
        border-radius: 2px;
        border: none;
        outline: none;
        height: 50px;
        font-size: 14px;
        color: #fff;
        cursor: pointer;
        text-transform: uppercase;
        position: relative;
        -webkit-transition: all 0.3s;
        transition: all 0.3s;
        padding-left: 30px;
        padding-right: 15px;
    }
    .icon {
        position: absolute;
        top: 30%;
        left: 10px;
        font-size: 20px;
```

```
}
.taskAdd {
    background: #444;
    padding-left: 31px;
}
.taskAdd:hover {
    background: #303030;
}
.taskDelete {
    background: #e74c3c;
    padding-left: 30px;
}
.taskDelete:hover {
    background: #c0392b;
}
/* task styles */
.taskList {
    list-style: none;
    padding: 0 20px;
}
.taskItem {
    border-top: 1px solid #e5e5e5;
    padding: 15px 0;
    color: #777;
    font-weight: 600;
    font-size: 14px;
    letter-spacing: 0.5px;
}
.taskList .taskItem:nth-child(even) {
    background: #fcfcfc;
}
.taskCheckbox {
    margin-right: 1em;
}
.complete-true {
    text-decoration:  line-through;
    color: #bebebe;
}
.taskList .taskDate {
    color: #95a5a6;
    font-size: 10px;
    font-weight: bold;
    text-transform: uppercase;
    display: block;
    margin-left: 41px;
```

```
}
.fa-calendar {
    margin-right: 10px;
    font-size: 16px;
}
[class*='category-'] {
    display: inline-block;
    font-size: 10px;
    background: #444;
    vertical-align: middle;
    color: #fff;
    padding: 10px;
    width: 75px;
    text-align: center;
    border-radius: 2px;
    float: right;
    font-weight: normal;
    text-transform: uppercase;
    margin-right: 20px;
}
.category- {
    background: transparent;
}
.category-Personal {
    background: #2980b9;
}
.category-Work {
    background: #8e44ad;
}
.category-School {
    background: #f39c12;
}
.category-Cleaning {
    background: #16a085;
}
.category-Other {
    background: #d35400;
}
footer {
    text-align: center;
    font-size: 11px;
    font-weight: 600;
    text-transform: uppercase;
    color: #777;
}
```

```
footer a {
    color: #f39c12;
}
/* custom checkboxes */
.taskCheckbox {
    -webkit-appearance: none;
            appearance: none;
    -webkit-transition: all 0.3s;
            transition: all 0.3s;
    display: inline-block;
    cursor: pointer;
    width: 19px;
    height: 19px;
    vertical-align: middle;
}
.taskCheckbox:focus {
    outline: none;
}
.taskCheckbox:before, .taskCheckbox:checked:before {
    font-family: 'FontAwesome';
    color: #444;
    font-size: 20px;
    -webkit-transition: all 0.3s;
            transition: all 0.3s;
}
.taskCheckbox:before {
    content: '\f096';
}
.taskCheckbox:checked:before {
    content: '\f14a';
    color: #16a085;
}
/* custom select menu */
.taskCategory {
    -webkit-appearance: none;
            appearance: none;
    cursor: pointer;
    padding-left: 16.5px; /*specific positioning due to difficult
behavior of select element*/
    background: #fff;
}
.selectArrow {
    position: absolute;
    z-index: 10;
    top: 35%;
```

```
        right: 0;
        margin-right: 20px;
        color: #777;
        pointer-events: none;
    }
    .taskCategory option {
        background: #fff;
        border: none;
        outline: none;
        padding: 0 100px;
    }
```

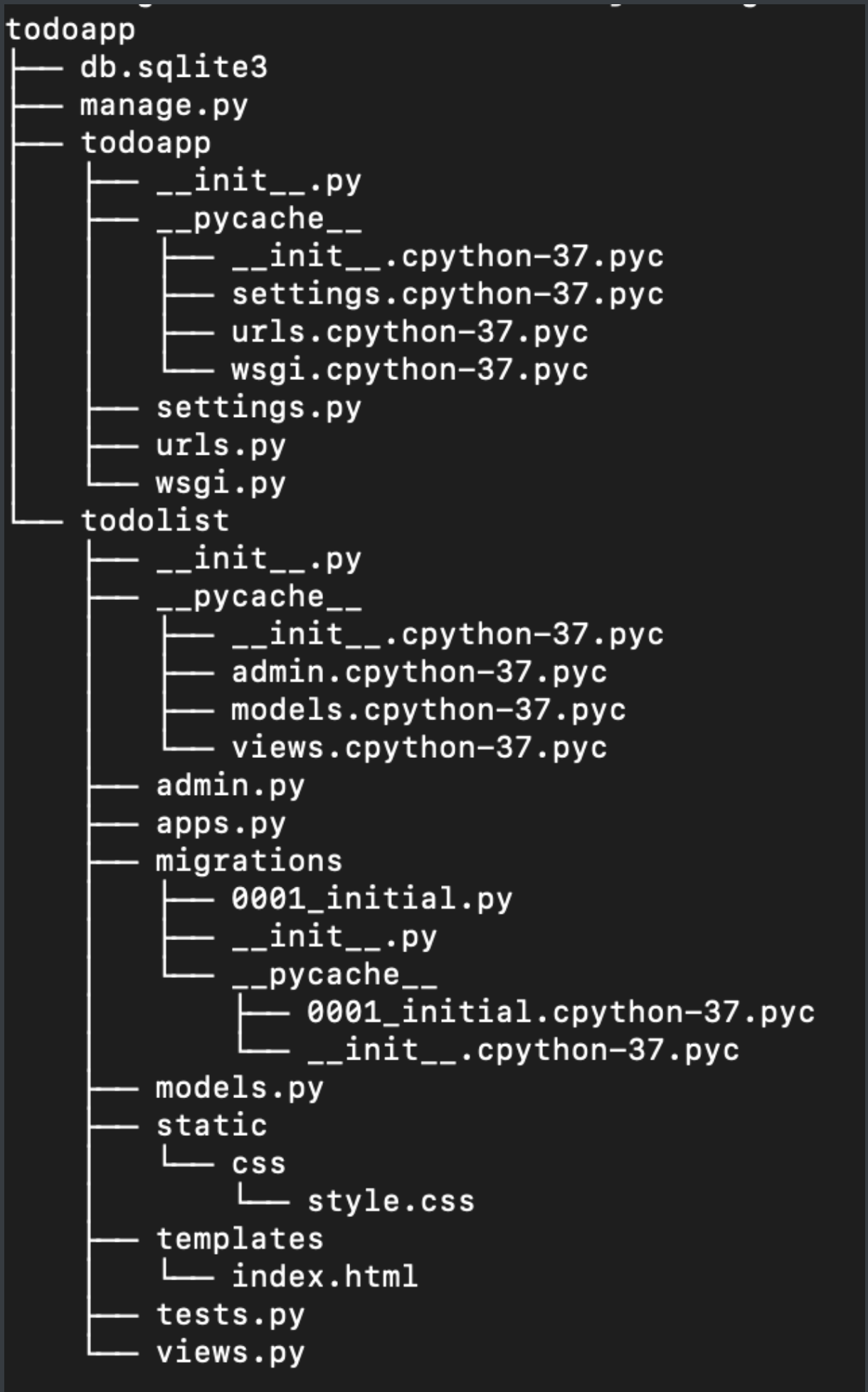
8. Update the urls.py

```
    from django.contrib import admin
    from django.urls import path
    from django.conf.urls import url
    from todolist.views import index

    urlpatterns = [
        path('admin/', admin.site.urls),
        url(r'^$', index, name="TodoList"),
    ]
```

9. Register the models in admin.py

```
    from django.contrib import admin
    from . import models

    # Register your models here.
    class TodoListAdmin(admin.ModelAdmin):
        list_display = ("title",  "created", "due_date")
    class CategoryAdmin(admin.ModelAdmin):
        list_display = ("name",)
    admin.site.register(models.TodoList, TodoListAdmin)
    admin.site.register(models.Category, CategoryAdmin)
```

10. FINISHED


todoapp
db.sqlite3
manage.py
todoapp
__init__.py
__pycache__
__init__.cpython-37.pyc
settings.cpython-37.pyc
urls.cpython-37.pyc
wsgi.cpython-37.pyc
settings.py
urls.py
wsgi.py
todolist
__init__.py
__pycache__
__init__.cpython-37.pyc
admin.cpython-37.pyc
models.cpython-37.pyc
views.cpython-37.pyc
admin.py
apps.py
migrations
0001_initial.py
__init__.py
__pycache__
0001_initial.cpython-37.pyc
__init__.cpython-37.pyc
models.py
static
css
style.css
templates
index.html
tests.py
views.py