**Setting up a Django development environment**

python.exe -m venv env

.\env\Scripts\activate

Installing packet

pip install django

create django project

django-admin startproject cliniqueplus

Creating the catalog application

py manage.py startapp utilisateur

py manage.py startapp core

Registering the catalog application

cliniqueplus\settings.py

# Application definition

INSTALLED\_APPS = [

    'django.contrib.admin',

    'django.contrib.auth',

    'django.contrib.contenttypes',

    'django.contrib.sessions',

    'django.contrib.messages',

    'django.contrib.staticfiles',

    # Add our new application

    'core.apps.CoreConfig', #This object was created for us in /core/apps.py

    'utilisateur.apps.UtilisateurConfig', #This object was created for us in /utilisateur/apps.py

]

Specifying the database

pip install psycopg2

cliniqueplus\settings.py

# Database

DATABASES = {

    'default': {

        'ENGINE': 'django.db.backends.postgresql',

        'NAME': 'dbname',

        'USER': 'dbuser',

        'PASSWORD': 'passsword',

        'HOST': 'dbhost',

        'PORT': '',

    }

}

Other project settings

# Internationalization

# https://docs.djangoproject.com/en/4.1/topics/i18n/

LANGUAGE\_CODE = 'fr-FR'

TIME\_ZONE = 'Africa/Lome'

Hooking up the URL mapper

core\urls.py

from django.urls import path

from . import views

urlpatterns = [

]

cliniqueplus\urls.py

from django.contrib import admin

from django.urls import path,include

from django.conf import settings

from django.conf.urls.static import static

urlpatterns = [

    path('admin/', admin.site.urls),

    path('', include('core.urls')),

    path('utilisateur/', include('utilisateur.urls')),

] + static(settings.STATIC\_URL, document\_root=settings.STATIC\_ROOT)

Running database migrations

python manage.py makemigrations

python manage.py migrate

Running the website

python manage.py runserver

Using models

Once we've decided on our models and field, we need to think about the relationships. Django allows you to define relationships that are one to one (OneToOneField), one to many (ForeignKey) and many to many (ManyToManyField).

La bibliotheque des images

pip install Pillow

django contri

pip install django-countries

phonenumber

pip install django-phonenumber-field[phonenumberslite]

django-taggit

pip install django-taggit

django-money

Installation

pip install django-money

Add djmoney to your INSTALLED\_APPS. This is required so that money field are displayed correctly in the admin.

INSTALLED\_APPS = [

   ...,

   'djmoney',

   ...

]

INSTALLED\_APPS = [

        # TIERS\_APPS

    'django\_countries',

    'phonenumber\_field',

 'taggit',

]

core\models.py

from django.db import models

from django.urls import reverse

import uuid

from django\_countries.fields import CountryField

from phonenumber\_field.modelfields import PhoneNumberField

from taggit.managers import TaggableManager

# Create your models here.

# la class de base

class BaseModel(models.Model):# toutes les autres class vont heriter de ces 3 attributs de la class BaseModel

    uid=models.UUIDField(primary\_key=True,default=uuid.uuid4,editable=False)

    status=models.BooleanField(default=True,help\_text='active ou desactive')

    created=models.DateField(auto\_now\_add=True,blank=True,null=True,help\_text='Create date')

    updated=models.DateTimeField(auto\_now=True,help\_text='Update date')

    class Meta:

        abstract=True

class Specialite(BaseModel):

    libelle=models.CharField(max\_length=255,blank=True,null=True)

    description = models.TextField(blank=True,null=True) #Short Description of the Specialite

    image = models.ImageField(upload\_to='specialites') #Cover Image of the Specialite

    # Metadata

    class Meta:

        ordering = ['-updated']

        verbose\_name = 'Specialite'

    def \_\_str\_\_(self):

        return self.libelle

    def get\_absolute\_url(self):

        return reverse("specialite\_details", kwargs={'pk': self.pk})

class Departement(BaseModel):

    libelle=models.CharField(max\_length=255,blank=True,null=True)

    description = models.TextField(blank=True,null=True) #Short Description of the Departement

    image = models.ImageField(upload\_to='departements') #Cover Image of the Departement

    def \_\_str\_\_(self):

        return self.libelle

    def get\_absolute\_url(self):

        return reverse("departement\_details", kwargs={'pk': self.pk})

class Ecole(BaseModel):

    libelle=models.CharField(max\_length=255,blank=True,null=True)

    description = models.TextField(blank=True,null=True) #Short Description of the Ecole

    def \_\_str\_\_(self):

        return self.libelle

    def get\_absolute\_url(self):

        return reverse("ecole\_details", kwargs={'pk': self.pk})

class Formation(BaseModel):

    libelle=models.CharField(max\_length=255,blank=True,null=True)

    description = models.TextField(blank=True,null=True) #Short Description of the Formation

    def \_\_str\_\_(self):

        return self.libelle

    def get\_absolute\_url(self):

        return reverse("formation\_details", kwargs={'pk': self.pk})

#   les champs d'action des medcins

class Specialisations(BaseModel):

    libelle=models.CharField(max\_length=255,blank=True,null=True)

    description = models.TextField(blank=True,null=True) #Short Description of the Specialisations

    def \_\_str\_\_(self):

        return self.libelle

    def get\_absolute\_url(self):

        return reverse("Specialisations\_details", kwargs={'pk': self.pk})

class Clinique(BaseModel):

    libelle=models.CharField(max\_length=255,blank=True,null=True)

    description = models.TextField(blank=True,null=True) #Short Description of the Clinique

    def \_\_str\_\_(self):

        return self.libelle

    def get\_absolute\_url(self):

        return reverse("clinique\_details", kwargs={'pk': self.pk})

class Medcin(BaseModel):

    GENRES = (

        ('h', 'Homme'),

        ('f', 'Femme'),

    )

    IDMedcin = models.CharField(max\_length=255,blank=True,null=True,unique=True)

    nom = models.CharField(max\_length=255,blank=True,null=True)

    prenom = models.CharField(max\_length=255,blank=True,null=True)

    email = models.EmailField(db\_index=True, unique=True)

    photo = models.ImageField(upload\_to='medcins/avatar/',blank=True, null=True)

    telephone = PhoneNumberField(null=False, blank=False, unique=True)

    pays = CountryField(null=True,blank=True,blank\_label='(select country)')

    adresse = models.CharField(max\_length=255,blank=True,null=True)

    bio = models.TextField(blank=True,null=True) #Short Description of the Specialite

    anniversaire = models.CharField(max\_length=255,blank=True,null=True)

    genre = models.CharField(

        max\_length=1,

        choices=GENRES,

        blank=True,

        default='h',

        help\_text='Select genre',

    )

    tags = TaggableManager()

    specialite=models.ForeignKey(Specialite,on\_delete=models.SET\_NULL,blank=True,null=True,related\_name="fk\_specialite")

    departement=models.ForeignKey(Departement,on\_delete=models.SET\_NULL,blank=True,null=True,related\_name="fk\_departement")

    views = models.IntegerField(default=0)

    is\_popular = models.BooleanField(default=False)

    # media social

    twitter = models.CharField(

        blank=True, null=True, name='twitter', help\_text="Twitter", max\_length=200)

    facebook = models.CharField(

        blank=True, null=True, name='facebook', help\_text="Facebook", max\_length=200)

    instagram = models.CharField(

        blank=True, null=True, name='instagram', help\_text="Instagram", max\_length=200)

    linkdin = models.CharField(

        blank=True, null=True, name='linkdin', help\_text="Linkdin", max\_length=200)

    youtube = models.CharField(

        blank=True, null=True, name='youtube', help\_text="Linkdin", max\_length=200)

    def \_\_str\_\_(self):

        return self.nom

    def get\_absolute\_url(self):

        return reverse("medcin\_details", kwargs={'pk': self.pk})

    def get\_shortname(self):

        return f'{self.prenom[0:1]}.{self.nom}'

    def get\_fullname(self):

        return f'{self.prenom} {self.nom}'

class Education(BaseModel):

    formation=models.CharField(max\_length=255,blank=True,null=True,help\_text='Le libelle de la formation')

    ecole=models.CharField(max\_length=255,blank=True,null=True,help\_text="Le libelle de l'cole")

    description = models.TextField(blank=True,null=True) #Short Description of the Education

    date\_debut = models.DateField(blank=True, null=True,help\_text='Date de debut de la formation')

    date\_fin = models.DateField(blank=True, null=True,help\_text='Date de fin de la formation')

    medcin = models.ForeignKey(Medcin,on\_delete=models.SET\_NULL,blank=True,null=True,help\_text='Selectionner le medcin',related\_name="fk\_medcin\_education")

    def \_\_str\_\_(self):

        return f'{self.formation} {self.medcin.nom}'

    def get\_absolute\_url(self):

        return reverse("education\_details", kwargs={'pk': self.pk})

class Experience(BaseModel):

    poste=models.CharField(max\_length=255,blank=True,null=True,help\_text='Le libelle du poste occupé')

    clinique = models.ForeignKey(Clinique,on\_delete=models.SET\_NULL,blank=True,null=True,help\_text='Selectionner la clinique',related\_name="fk\_clinique\_experience")

    description = models.TextField(blank=True,null=True) #Short Description of the Experience

    date\_debut = models.DateField(blank=True, null=True,help\_text='Date de debut de la formation')

    date\_fin = models.DateField(blank=True, null=True,help\_text='Date de fin de la formation')

    medcin = models.ForeignKey(Medcin,on\_delete=models.SET\_NULL,blank=True,null=True,help\_text='Selectionner le medcin',related\_name="fk\_clinique\_experience")

    def \_\_str\_\_(self):

        return f'{self.poste} {self.medcin.nom}'

    def get\_absolute\_url(self):

        return reverse("experience\_details", kwargs={'pk': self.pk})

class Certificat(BaseModel):

    titre=models.CharField(max\_length=255,blank=True,null=True,help\_text='Le libelle du Certificat')

    description = models.TextField(blank=True,null=True) #Short Description of the Certificat

    date = models.DateField(blank=True, null=True,help\_text='Date')

    medcin = models.ForeignKey(Medcin,on\_delete=models.SET\_NULL,blank=True,null=True,help\_text='Selectionner le medcin',related\_name="fk\_Certificat")

    def \_\_str\_\_(self):

        return f'{self.titre} {self.medcin.nom}'

    def get\_absolute\_url(self):

        return reverse("certificat\_details", kwargs={'pk': self.pk})

class Galerie(BaseModel):

    titre=models.CharField(max\_length=255,blank=True,null=True,help\_text="Le libelle de l'cole")

    photo = models.ImageField(upload\_to='medcins/galerie/',blank=True, null=True)

    description = models.TextField(blank=True,null=True) #Short Description of the Galerie

    medcin = models.ForeignKey(Medcin,on\_delete=models.SET\_NULL,blank=True,null=True,help\_text='Selectionner le medcin',related\_name="fk\_medcin\_galerie")

    def \_\_str\_\_(self):

        return f'{self.titre} {self.medcin.nom}'

    def get\_absolute\_url(self):

        return reverse("galerie\_details", kwargs={'pk': self.pk})

class Disponibilite(BaseModel):

    joursDisponibilite=models.CharField(max\_length=255,blank=True,null=True,help\_text='Les jours de disponibilité')

    tempsDisponibilite=models.CharField(max\_length=255,blank=True,null=True,help\_text="Temps de disponibilité")

    medcin = models.ForeignKey(Medcin,on\_delete=models.SET\_NULL,blank=True,null=True,help\_text='Selectionner le medcin',related\_name="fk\_medcin\_disponibilite")

    description = models.TextField(blank=True,null=True,help\_text='Ce champ est ptionnel') #Short Description of the Disponibilite

    # lundi

    heure\_debut\_lundi = models.TimeField(blank=True, null=True,help\_text='Heure de debut')

    heure\_fin\_lundi = models.TimeField(blank=True, null=True,help\_text='Heure de fin de fin fin')

    # mardi

    heure\_debut\_mardi = models.TimeField(blank=True, null=True,help\_text='Heure de debut')

    heure\_fin\_mardi = models.TimeField(blank=True, null=True,help\_text='Heure de fin de fin')

    # mercredi

    heure\_debut\_mercredi = models.TimeField(blank=True, null=True,help\_text='Heure de debut')

    heure\_fin\_mercredi = models.TimeField(blank=True, null=True,help\_text='Heure de fin de fin')

    # jeudi

    heure\_debut\_jeudi = models.TimeField(blank=True, null=True,help\_text='Heure de debut')

    heure\_fin\_jeudi = models.TimeField(blank=True, null=True,help\_text='Heure de fin de fin')

    # vendredi

    heure\_debut\_vendredi = models.TimeField(blank=True, null=True,help\_text='Heure de debut')

    heure\_fin\_vendredi = models.TimeField(blank=True, null=True,help\_text='Heure de fin de fin')

    # samedi

    heure\_debut\_samedi = models.TimeField(blank=True, null=True,help\_text='Heure de debut')

    heure\_fin\_samedi = models.TimeField(blank=True, null=True,help\_text='Heure de fin de fin')

    # dimanche

    heure\_debut\_dimanche = models.TimeField(blank=True, null=True,help\_text='Heure de debut')

    heure\_fin\_dimanche = models.TimeField(blank=True, null=True,help\_text='Heure de fin de fin')

    def \_\_str\_\_(self):

        return f'{self.medcin.nom}'

    def get\_absolute\_url(self):

        return reverse("disponibilite\_details", kwargs={'pk': self.pk})

class SpecialisationMedcin(BaseModel):

    libelle=models.CharField(max\_length=255,blank=True,null=True)

    description = models.TextField(blank=True,null=True) #Short Description of the SpecialisationMedcin

    medcin = models.ForeignKey(Medcin,on\_delete=models.SET\_NULL,blank=True,null=True,help\_text='Selectionner le medcin',related\_name="fk\_SpecialisationMedcin")

    def \_\_str\_\_(self):

        return self.libelle

    def get\_absolute\_url(self):

        return reverse("SpecialisationsMedcin\_details", kwargs={'pk': self.pk})

Re-run the database migrations

python manage.py makemigrations

python manage.py migrate

Django admin site

Django exporte

pip install django-import-export

Registering models

core\admin.py

from django.contrib import admin

from .models import \*

from import\_export.admin import ImportExportModelAdmin

from django import forms

# Register your models here.

@admin.register(Specialite)

class SpecialiteAdmin(ImportExportModelAdmin, admin.ModelAdmin):

    list\_display=('libelle', 'description', )

@admin.register(Departement)

class DepartementAdmin(ImportExportModelAdmin, admin.ModelAdmin):

    list\_display=('libelle', 'description', )

@admin.register(Ecole)

class EcoleAdmin(ImportExportModelAdmin, admin.ModelAdmin):

    list\_display=('libelle', 'description', )

@admin.register(Formation)

class FormationAdmin(ImportExportModelAdmin, admin.ModelAdmin):

    list\_display=('libelle', 'description', )

@admin.register(Specialisations)

class SpecialisationsAdmin(ImportExportModelAdmin, admin.ModelAdmin):

    list\_display=('libelle', 'description', )

@admin.register(Clinique)

class CliniqueAdmin(ImportExportModelAdmin, admin.ModelAdmin):

    list\_display=('libelle', 'description', )

class EduAdmin(admin.StackedInline):

    model= Education

class ExpAdmin(admin.StackedInline):

    model= Experience

# @admin.register(Medcin)

class MedcinAdmin(ImportExportModelAdmin, admin.ModelAdmin):

    list\_display = ('IDMedcin', 'nom', 'prenom', 'email', 'photo', 'telephone', 'pays', 'adresse', 'bio', 'anniversaire', 'genre', 'tags', 'specialite', 'views', 'is\_popular', 'twitter', 'facebook', 'instagram', 'linkdin', )

    ordering=('updated',)

    inlines=[EduAdmin,ExpAdmin]

    list\_filter = ('status', 'genre','specialite','departement')

admin.site.register(Medcin,MedcinAdmin)

@admin.register(Education)

class EducationAdmin(ImportExportModelAdmin, admin.ModelAdmin):

    list\_display=('formation', 'description', 'date\_debut', 'date\_fin', )

@admin.register(Experience)

class ExperienceAdmin(ImportExportModelAdmin, admin.ModelAdmin):

    list\_display=('poste', 'description', 'date\_debut', 'date\_fin', )

@admin.register(Certificat)

class CertificatAdmin(ImportExportModelAdmin, admin.ModelAdmin):

    list\_display=('titre', 'date', 'medcin', )

@admin.register(Galerie)

class GalerieAdmin(ImportExportModelAdmin, admin.ModelAdmin):

    list\_display=('titre', 'description', 'medcin')

@admin.register(Disponibilite)

class DisponibiliteAdmin(ImportExportModelAdmin, admin.ModelAdmin):

    list\_display=('joursDisponibilite', 'medcin', 'description', 'heure\_debut\_lundi', 'heure\_fin\_lundi', 'heure\_debut\_mardi', 'heure\_fin\_mardi', 'heure\_debut\_mercredi', 'heure\_fin\_mercredi', 'heure\_debut\_jeudi', 'heure\_fin\_jeudi', 'heure\_debut\_vendredi', 'heure\_fin\_vendredi', 'heure\_debut\_samedi', 'heure\_fin\_samedi', 'heure\_debut\_dimanche', )

@admin.register(SpecialisationMedcin)

class SpecialisationMedcinAdmin(ImportExportModelAdmin, admin.ModelAdmin):

    list\_display=('libelle', 'medcin', 'description',)

Creating a superuser

python manage.py createsuperuser

Django REST framework

Django REST framework is a powerful and flexible toolkit for building Web APIs.

## [**Installation**](https://www.django-rest-framework.org/#installation)

Install using pip, including any optional packages you want...

pip install djangorestframework

pip install markdown # Markdown support for the browsable API.

pip install django-filter # Filtering support

pip install pygments # We'll be using this for the code highlighting

Add 'rest\_framework' to your INSTALLED\_APPS setting.

INSTALLED\_APPS = [

...

'rest\_framework',

]

If you're intending to use the browsable API you'll probably also want to add REST framework's login and logout views. Add the following to your root urls.py file.

urlpatterns = [

...

path('api-auth/', include('rest\_framework.urls'))

]

We'll create a read-write API for accessing information on the users of our project.

Any global settings for a REST framework API are kept in a single configuration dictionary named REST\_FRAMEWORK. Start off by adding the following to your settings.py module:

REST\_FRAMEWORK = {

    # Use Django's standard `django.contrib.auth` permissions,

    # or allow read-only access for unauthenticated users.

    'DEFAULT\_PERMISSION\_CLASSES': [

        'rest\_framework.permissions.DjangoModelPermissionsOrAnonReadOnly'

    ]

}

Serialization

# Django REST Libs:

from rest\_framework import serializers

# Local Libs:

from .models import BaseModel, Specialite, Departement, Ecole, Formation, Specialisations, Clinique, Medcin, Education, Experience, Certificat, Galerie, Disponibilite, SpecialisationMedcin, Pays, Region, Ville, Profession, Patient

class BaseModelSerializer(serializers.ModelSerializer):

    """

    BaseModel serializer

    Based on serializers.ModelSerializer

    """

    class Meta:

        model = BaseModel

        fields = "\_\_all\_\_"

class SpecialiteSerializer(serializers.ModelSerializer):

    """

    Specialite serializer

    Based on serializers.ModelSerializer

    """

    class Meta:

        model = Specialite

        fields = "\_\_all\_\_"

class DepartementSerializer(serializers.ModelSerializer):

    """

    Departement serializer

    Based on serializers.ModelSerializer

    """

    class Meta:

        model = Departement

        fields = "\_\_all\_\_"

class EcoleSerializer(serializers.ModelSerializer):

    """

    Ecole serializer

    Based on serializers.ModelSerializer

    """

    class Meta:

        model = Ecole

        fields = "\_\_all\_\_"

class FormationSerializer(serializers.ModelSerializer):

    """

    Formation serializer

    Based on serializers.ModelSerializer

    """

    class Meta:

        model = Formation

        fields = "\_\_all\_\_"

class SpecialisationsSerializer(serializers.ModelSerializer):

    """

    Specialisations serializer

    Based on serializers.ModelSerializer

    """

    class Meta:

        model = Specialisations

        fields = "\_\_all\_\_"

class CliniqueSerializer(serializers.ModelSerializer):

    """

    Clinique serializer

    Based on serializers.ModelSerializer

    """

    class Meta:

        model = Clinique

        fields = "\_\_all\_\_"

class MedcinSerializer(serializers.ModelSerializer):

    """

    Medcin serializer

    Based on serializers.ModelSerializer

    """

    class Meta:

        model = Medcin

        fields = "\_\_all\_\_"

class EducationSerializer(serializers.ModelSerializer):

    """

    Education serializer

    Based on serializers.ModelSerializer

    """

    class Meta:

        model = Education

        fields = "\_\_all\_\_"

class ExperienceSerializer(serializers.ModelSerializer):

    """

    Experience serializer

    Based on serializers.ModelSerializer

    """

    class Meta:

        model = Experience

        fields = "\_\_all\_\_"

class CertificatSerializer(serializers.ModelSerializer):

    """

    Certificat serializer

    Based on serializers.ModelSerializer

    """

    class Meta:

        model = Certificat

        fields = "\_\_all\_\_"

class GalerieSerializer(serializers.ModelSerializer):

    """

    Galerie serializer

    Based on serializers.ModelSerializer

    """

    class Meta:

        model = Galerie

        fields = "\_\_all\_\_"

class DisponibiliteSerializer(serializers.ModelSerializer):

    """

    Disponibilite serializer

    Based on serializers.ModelSerializer

    """

    class Meta:

        model = Disponibilite

        fields = "\_\_all\_\_"

class SpecialisationMedcinSerializer(serializers.ModelSerializer):

    """

    SpecialisationMedcin serializer

    Based on serializers.ModelSerializer

    """

    class Meta:

        model = SpecialisationMedcin

        fields = "\_\_all\_\_"

class PaysSerializer(serializers.ModelSerializer):

    """

    Pays serializer

    Based on serializers.ModelSerializer

    """

    class Meta:

        model = Pays

        fields = "\_\_all\_\_"

class RegionSerializer(serializers.ModelSerializer):

    """

    Region serializer

    Based on serializers.ModelSerializer

    """

    class Meta:

        model = Region

        fields = "\_\_all\_\_"

class VilleSerializer(serializers.ModelSerializer):

    """

    Ville serializer

    Based on serializers.ModelSerializer

    """

    class Meta:

        model = Ville

        fields = "\_\_all\_\_"

class ProfessionSerializer(serializers.ModelSerializer):

    """

    Profession serializer

    Based on serializers.ModelSerializer

    """

    class Meta:

        model = Profession

        fields = "\_\_all\_\_"

class PatientSerializer(serializers.ModelSerializer):

    """

    Patient serializer

    Based on ser

ializers.ModelSerializer

    """

    class Meta:

        model = Patient

        fields = "\_\_all\_\_"

django and react

Putting in the APIs

Install the djangorestframework and django-cors-headers:

pip install djangorestframework django-cors-headers

Add rest\_framework and corsheaders to the INSTALLED\_APPS in backend/settings.py file and modifiy the MIDDLEWARE:

# Application definition

INSTALLED\_APPS = [

    'django.contrib.admin',

    'django.contrib.auth',

    'django.contrib.contenttypes',

    'django.contrib.sessions',

    'django.contrib.messages',

    'django.contrib.staticfiles',

    # TIERS\_APPS

    'corsheaders',

    'rest\_framework',

    'rest\_framework.authtoken',

]

MIDDLEWARE = [

    'corsheaders.middleware.CorsMiddleware', #new

    'django.middleware.security.SecurityMiddleware',

    'django.contrib.sessions.middleware.SessionMiddleware',

    'django.middleware.common.CommonMiddleware',

    'django.middleware.csrf.CsrfViewMiddleware',

    'django.contrib.auth.middleware.AuthenticationMiddleware',

    'django.contrib.messages.middleware.MessageMiddleware',

    'django.middleware.clickjacking.XFrameOptionsMiddleware',

]

Add this code snippet in backend/settings.py file:

CORS\_ORIGIN\_WHITELIST = [

    'http://localhost:3000',

]

### **Frontend using React**

To install React we use the following command:

-g stands for global as we are first installing create-react-app globally:

npm install -g create-react-app

While in the parent directory - django-react-todoApp- create a React application, frontend:

create-react-app ui\_clinique

To start the server:

cd frontend

npm start

Next, let’s install bootstrap and reactstrap to style the user interface

npm install bootstrap@4.6.0 reactstrap@8.9.0 --legacy-peer-deps

Extension vscode

Django factory

Django

**Simple React Snippets**v

## **Snippets**

| **Snippet** | **Renders** |
| --- | --- |
| imr | Import React |
| imrc | Import React / Component |
| imrd | Import ReactDOM |
| imrs | Import React / useState |
| imrse | Import React / useState useEffect |
| impt | Import PropTypes |
| impc | Import React / PureComponent |
| cc | Class Component |
| ccc | Class Component With Constructor |
| cpc | Class Pure Component |
| ffc | Function Component |
| sfc | Stateless Function Component (Arrow function) |
| cdm | componentDidMount |
| uef | useEffect Hook |
| cwm | componentWillMount |
| cwrp | componentWillReceiveProps |
| gds | getDerivedStateFromProps |
| scu | shouldComponentUpdate |
| cwu | componentWillUpdate |
| cdu | componentDidUpdate |
| cwun | componentWillUnmount |
| cdc | componentDidCatch |
| gsbu | getSnapshotBeforeUpdate |
| ss | setState |
| ssf | Functional setState |
| usf | Declare a new state variable using State Hook |
| ren | render |
| rprop | Render Prop |
| hoc | Higher Order Component |
| cp | Context Provider |
| cpf | Class Property Function |

## **Full Expansions**

### **imr - Import React**

import \* as React from "react";

### **imrc - Import React, Component**

import \* as React from "react";  
import { Component } from "react";

### **imrd - Import ReactDOM**

import ReactDOM from "react-dom";

### **imrs - Import React, useState**

import \* as React from "react";  
import { useState } from "react";

### **imrse - Import React, useState, useEffect**

import \* as React from "react";  
import { useState, useEffect } from "react";

### **impt - Import PropTypes**

import PropTypes from "prop-types";

### **impc - Import PureComponent**

import \* as React from "react";  
import { PureComponent } from "react";

### **cc - Class Component**

class | extends React.Component {  
 render() {  
 return <div>|</div>  
 }  
}  
  
export default |;

### **ccc - Class Component With Constructor**

class | extends Component {  
 constructor(props) {  
 super(props);  
 this.state = { | };  
 }  
 render() {  
 return ( | );  
 }  
}  
  
export default |;

### **cpc - Class Pure Component**

class | extends PureComponent {  
 state = { | },  
 render() {  
 return ( | );  
 }  
}  
  
export default |;

### **ffc - Function Component**

function (|) {  
 return ( | );  
}  
  
export default |;

### **sfc - Stateless Function Component (Arrow function)**

const | = props => {  
 return ( | );  
};  
  
export default |;

### **cdm - componentDidMount**

componentDidMount() {  
 |  
}

### **uef - useEffect Hook**

useEffect(() => {  
 |  
}, []);

### **cwm - componentWillMount**

//WARNING! To be deprecated in React v17. Use componentDidMount instead.  
componentWillMount() {  
 |  
}

### **cwrp - componentWillReceiveProps**

//WARNING! To be deprecated in React v17. Use new lifecycle static getDerivedStateFromProps instead.  
componentWillReceiveProps(nextProps) {  
 |  
}

### **gds - getDerivedStateFromProps**

static getDerivedStateFromProps(nextProps, prevState) {  
 |  
}

### **scu - shouldComponentUpdate**

shouldComponentUpdate(nextProps, nextState) {  
 |  
}

### **cwu - componentWillUpdate**

//WARNING! To be deprecated in React v17. Use componentDidUpdate instead.  
componentWillUpdate(nextProps, nextState) {  
 |  
}

### **cdu - componentDidUpdate**

componentDidUpdate(prevProps, prevState) {  
 |  
}

### **cwun - componentWillUnmount**

componentWillUnmount() {  
 |  
}

### **cdc - componentDidCatch**

componentDidCatch(error, info) {  
 |  
}

### **gsbu - getSnapshotBeforeUpdate**

getSnapshotBeforeUpdate(prevProps, prevState) {  
 |  
}

### **ss - setState**

this.setState({ | : | });

### **ssf - Functional setState**

this.setState(prevState => {  
 return { | : prevState.| }  
});

### **usf - Declare a new state variable using State Hook**

const [|, set|] = useState();

Hit Tab to apply CamelCase to function. e.g. [count, setCount]

### **ren - render**

render() {  
 return (  
 |  
 );  
}

### **rprop - Render Prop**

class | extends Component {  
 state = { | },  
 render() {  
 return this.props.render({  
 |: this.state.|  
 });  
 }  
}  
  
export default |;

### **hoc - Higher Order Component**

function | (|) {  
 return class extends Component {  
 constructor(props) {  
 super(props);  
 }  
  
 render() {  
 return < | {...this.props} />;  
 }  
 };  
}

### **cpf - Class Property Function**

| = (e) => {  
 |  
 }

## **Commands**

### **React: class to className**

Changes all occurences of class in JSX to className. This transform is safe to run multiple times on any document. No text needs to be selected as the command is executed on the entire document.

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