# RURAL CARE Project: Backend Implementation using Python, Django, and PostgreSQL

# Install dependencies

# pip install django djangorestframework psycopg2 twilio boto3

# settings.py (Django settings for AWS and PostgreSQL integration)

DATABASES = {

'default': {

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

'NAME': 'rural\_care',

'USER': 'your\_postgres\_user',

'PASSWORD': 'your\_postgres\_password',

'HOST': 'localhost',

'PORT': '5432',

}

}

# AWS S3 for storage

AWS\_ACCESS\_KEY\_ID = 'your\_aws\_access\_key'

AWS\_SECRET\_ACCESS\_KEY = 'your\_aws\_secret\_key'

AWS\_STORAGE\_BUCKET\_NAME = 'your\_bucket\_name'

AWS\_S3\_REGION\_NAME = 'your\_region'

# Twilio settings

TWILIO\_ACCOUNT\_SID = 'your\_twilio\_account\_sid'

TWILIO\_AUTH\_TOKEN = 'your\_twilio\_auth\_token'

# models.py

from django.db import models

class Patient(models.Model):

name = models.CharField(max\_length=100)

age = models.IntegerField()

address = models.TextField()

phone\_number = models.CharField(max\_length=15)

health\_condition = models.TextField()

def \_\_str\_\_(self):

return self.name

class Appointment(models.Model):

patient = models.ForeignKey(Patient, on\_delete=models.CASCADE)

doctor\_name = models.CharField(max\_length=100)

appointment\_time = models.DateTimeField()

def \_\_str\_\_(self):

return f"{self.patient.name} - {self.doctor\_name}"

# serializers.py

from rest\_framework import serializers

from .models import Patient, Appointment

class PatientSerializer(serializers.ModelSerializer):

class Meta:

model = Patient

fields = '\_\_all\_\_'

class AppointmentSerializer(serializers.ModelSerializer):

class Meta:

model = Appointment

fields = '\_\_all\_\_'

# views.py

from rest\_framework.views import APIView

from rest\_framework.response import Response

from rest\_framework import status

from .models import Patient, Appointment

from .serializers import PatientSerializer, AppointmentSerializer

class PatientView(APIView):

def get(self, request):

patients = Patient.objects.all()

serializer = PatientSerializer(patients, many=True)

return Response(serializer.data)

def post(self, request):

serializer = PatientSerializer(data=request.data)

if serializer.is\_valid():

serializer.save()

return Response(serializer.data, status=status.HTTP\_201\_CREATED)

return Response(serializer.errors, status=status.HTTP\_400\_BAD\_REQUEST)

class AppointmentView(APIView):

def get(self, request):

appointments = Appointment.objects.all()

serializer = AppointmentSerializer(appointments, many=True)

return Response(serializer.data)

def post(self, request):

serializer = AppointmentSerializer(data=request.data)

if serializer.is\_valid():

serializer.save()

return Response(serializer.data, status=status.HTTP\_201\_CREATED)

return Response(serializer.errors, status=status.HTTP\_400\_BAD\_REQUEST)

# urls.py

from django.urls import path

from .views import PatientView, AppointmentView

urlpatterns = [

path('patients/', PatientView.as\_view(), name='patients'),

path('appointments/', AppointmentView.as\_view(), name='appointments'),

]

# Example usage of Twilio for communication (e.g., appointment reminders)

from twilio.rest import Client

def send\_appointment\_reminder(phone\_number, message):

client = Client(TWILIO\_ACCOUNT\_SID, TWILIO\_AUTH\_TOKEN)

client.messages.create(

to=phone\_number,

from\_="your\_twilio\_phone\_number",

body=message

)

# Example: Using AWS S3 to upload patient files

import boto3

def upload\_file\_to\_s3(file\_name, bucket, object\_name=None):

s3\_client = boto3.client('s3',

aws\_access\_key\_id=AWS\_ACCESS\_KEY\_ID,

aws\_secret\_access\_key=AWS\_SECRET\_ACCESS\_KEY)

try:

s3\_client.upload\_file(file\_name, bucket, object\_name or file\_name)

except Exception as e:

print(f"Error uploading to S3: {e}")

# Optimizing Android apps for offline and low-resource environments

# Implement offline-first design in Android apps by using Room Database for local data storage.

# This ensures the app works without internet and syncs with the backend when connectivity is restored.

# Example: Android Room Database Schema (Java/Kotlin Example)

# @Entity

# data class Patient(

# @PrimaryKey val id: Int,

# val name: String,

# val age: Int,

# val address: String,

# val phoneNumber: String,

# val healthCondition: String

# )

# Example: Data Sync Logic (Pseudocode)

# - Save patient data locally using Room.

# - Use a WorkManager to periodically sync data with the backend when connectivity is available.

#

# val workRequest = OneTimeWorkRequestBuilder<DataSyncWorker>()

# .setConstraints(Constraints.Builder()

# .setRequiredNetworkType(NetworkType.CONNECTED)

# .build())

# .build()

# WorkManager.getInstance(context).enqueue(workRequest)

# Run Django migrations and start the development server

# python manage.py makemigrations

# python manage.py migrate

# python manage.py runserver