```
from django.db import models
from django.contrib.auth.models import AbstractUser
from django.utils import timezone

class User(AbstractUser):
    ROLE_CHOICES = [
        ('student', 'Student'),
        ('parent', 'Parent'),
        ('teacher', 'Teacher'),
        ('staff', 'Staff'),
        ('headteacher', 'Headteacher'),
        ('admin', 'Admin'),
    ]
    role = models.CharField(max_length=20, choices=ROLE_CHOICES, default='student')
    language_preference = models.CharField(max_length=5, default='en')  # 'en' or 'ur'

    def save(self, *args, **kwargs):
        # set default password '15586' only when creating a new user and no password provided
        if not self.pk and not self.password:
            self.set_password('15586')
        super().save(*args, **kwargs)

    def __str__(self):
        return f"{self.username} ({self.role})"

class Student(models.Model):
    user = models.OneToOneField(User, on_delete=models.CASCADE,
related_name="student_profile")
    class_name = models.CharField(max_length=50, blank=True)

    def __str__(self):
        return f"{self.user.get_full_name() or self.user.username} - {self.class_name}"

class Attendance(models.Model):
    STATUS_CHOICES = [
        ('Present','Present'),
        ('Absent','Absent'),
        ('Late','Late'),
        ('Custom','Custom'),
    ]
    student = models.ForeignKey(Student, on_delete=models.CASCADE,
related_name="attendances")
    date = models.DateField(default=timezone.localdate)
    status = models.CharField(max_length=20, choices=STATUS_CHOICES)
    reason = models.CharField(max_length=200, blank=True)

    class Meta:
        ordering = ['-date']

class Behaviour(models.Model):
    """
    category values:
    1 Excellent
    2 Good
    3 Disruptive
    4 Poor
    5 Physical Conflict
    """
    student = models.ForeignKey(Student, on_delete=models.CASCADE,
related_name="behaviours")
    category = models.IntegerField()
    custom_name = models.CharField(max_length=200, blank=True)
    date = models.DateField(default=timezone.localdate)
    detention_triggered = models.BooleanField(default=False)

    class Meta:
        ordering = ['-date']

class Payment(models.Model):
    student = models.ForeignKey(Student, on_delete=models.CASCADE,
related_name="payments")
    amount = models.FloatField()
    date = models.DateField(default=timezone.localdate)
    method = models.CharField(max_length=100, blank=True)  # JazzCash, Easypaisa, Bank
    status = models.CharField(max_length=20, default='Pending')  # Paid, Pending

    class Meta:
        ordering = ['-date']
```

school_system/admin.py
Put this in backend/school_system/admin.py

python
Copy code

```
from django.contrib import admin
from django.contrib.auth.admin import UserAdmin as BaseUserAdmin
from .models import User, Student, Attendance, Behaviour, Payment

@admin.register(User)
```

```
class UserAdmin(BaseUserAdmin):
    fieldsets = BaseUserAdmin.fieldsets + (
        ("Extra", {"fields": ("role", "language_preference")}),
    )

@admin.register(Student)
class StudentAdmin(admin.ModelAdmin):
    list_display = ("user", "class_name")

@admin.register(Attendance)
class AttendanceAdmin(admin.ModelAdmin):
    list_display = ("student", "date", "status", "reason")

@admin.register(Behaviour)
class BehaviourAdmin(admin.ModelAdmin):
    list_display = ("student", "date", "category", "custom_name", "detention_triggered")

@admin.register(Payment)
class PaymentAdmin(admin.ModelAdmin):
    list_display = ("student", "amount", "date", "method", "status")
```

school_system/serializers.py
Put this in backend/school_system/serializers.py

python
Copy code

```
from rest_framework import serializers
from .models import Behaviour, Attendance, Student, Payment, User

class UserSerializer(serializers.ModelSerializer):
    class Meta:
        model = User
        fields = ("id","username","first_name","last_name","email","role","language_preference")

class StudentSerializer(serializers.ModelSerializer):
    user = UserSerializer()
    class Meta:
        model = Student
        fields = ("id","user","class_name")

class BehaviourSerializer(serializers.ModelSerializer):
    class Meta:
        model = Behaviour
        fields = ("id","student","category","custom_name","date","detention_triggered")
```

```
class AttendanceSerializer(serializers.ModelSerializer):
    class Meta:
        model = Attendance
        fields = ("id","student","date","status","reason")

class PaymentSerializer(serializers.ModelSerializer):
    class Meta:
        model = Payment
        fields = ("id","student","amount","date","method","status")
```
school_system/utils.py
Put this in backend/school_system/utils.py

python
Copy code
```
import os
from gtts import gTTS
from django.conf import settings
from pathlib import Path
import uuid

VOICE_FOLDER = Path(settings.BASE_DIR) / "voice_files"
VOICE_FOLDER.mkdir(parents=True, exist_ok=True)

def voice_alert(message: str, language: str = 'ur') -> str:
    """
    Generate a voice MP3 for the message. Returns a relative file URL path.
    language: 'ur' or 'en'
    """
    # sanitize language
    lang = 'ur' if language == 'ur' else 'en'
    filename = f"alert_{uuid.uuid4().hex}.mp3"
    filepath = VOICE_FOLDER / filename
    tts = gTTS(text=message, lang=lang)
    tts.save(str(filepath))
    # return a path the frontend can request (if serving static files) or a filesystem path
    return str(filepath)
```
school_system/logic.py
Put this in backend/school_system/logic.py

python
Copy code
```
from .models import Behaviour
from .utils import voice_alert
```

```
def check_behaviour_trigger(student):
    """
    Check the student's behaviour records. If there are 2 or more 'Poor' (category==4) entries,
    mark the latest behaviour's detention_triggered True, generate a voice alert, and return
details.
    """
    behaviours = list(Behaviour.objects.filter(student=student).order_by('-date'))
    poor_count = sum(1 for b in behaviours if b.category == 4)

    if poor_count >= 2:
        # mark latest Behaviour record's detention flag
        latest = behaviours[0]  # because ordered by -date
        latest.detention_triggered = True
        latest.save()

        # prepare a message in the student's user language
        lang = student.user.language_preference
        if lang == 'ur':
            message = f"{student.user.get_full_name() or student.user.username} کو دو 'Poor' نمبرز
ملے ہیں، ڈیٹینشن کے لیے نشان زد۔"
        else:
            message = f"{student.user.get_full_name() or student.user.username} has received two
'Poor' marks and is flagged for detention."

        voice_file = voice_alert(message, lang)
        return {"triggered": True, "message": message, "voice_file": voice_file}
    return {"triggered": False}
```

school_system/views.py

Put this in backend/school_system/views.py — simple API views for creating behaviour and attendance

python

Copy code

```
from rest_framework.views import APIView
from rest_framework.response import Response
from rest_framework import status
from .serializers import BehaviourSerializer, AttendanceSerializer
from .models import Student
from .logic import check_behaviour_trigger

class BehaviourCreateView(APIView):
    """
    POST JSON:
    {
      "student": <student_id>,
      "category": 4,
      "custom_name": "Late to class"
    }
    """
    def post(self, request):
        serializer = BehaviourSerializer(data=request.data)
        if serializer.is_valid():
            behaviour = serializer.save()
            try:
                student = Student.objects.get(id=request.data.get('student'))
            except Student.DoesNotExist:
                return Response({"error":"Student not found"},
status=status.HTTP_400_BAD_REQUEST)

            result = check_behaviour_trigger(student)
            # include latest behaviour id to front-end
            result['behaviour_id'] = behaviour.id
            return Response(result, status=status.HTTP_201_CREATED)
        return Response(serializer.errors, status=status.HTTP_400_BAD_REQUEST)

class AttendanceCreateView(APIView):
    """
    POST JSON:
    {
      "student": <student_id>,
      "status": "Absent",
      "reason": "Sick"
    }
    """
    def post(self, request):
        serializer = AttendanceSerializer(data=request.data)
        if serializer.is_valid():
            att = serializer.save()
            return Response({"saved": True, "attendance_id": att.id},
status=status.HTTP_201_CREATED)
        return Response(serializer.errors, status=status.HTTP_400_BAD_REQUEST)
```

school_system/urls.py
Put this in backend/school_system/urls.py

python
Copy code

```
from django.urls import path
from .views import BehaviourCreateView, AttendanceCreateView
```

```
urlpatterns = [
    path("behaviour/", BehaviourCreateView.as_view(), name="behaviour-create"),
    path("attendance/", AttendanceCreateView.as_view(), name="attendance-create"),
]
```

How to run (quick)

Create project folder SchoolManagementSystem/backend/ and place the files above in the correct subfolders:

manage.py at backend/

school_management/ folder with settings.py, urls.py, wsgi.py

school_system/ app folder with models.py, admin.py, serializers.py, views.py, urls.py, utils.py, logic.py

requirements.txt in backend/

From terminal:

bash

Copy code

```
cd SchoolManagementSystem/backend

# create venv (recommended)
python -m venv venv
# activate it:
# macOS/Linux:
source venv/bin/activate
# Windows (PowerShell):
# .\venv\Scripts\Activate.ps1

pip install -r requirements.txt

# create Django app migrations
python manage.py makemigrations
python manage.py migrate

# create admin user to access Django admin UI
python manage.py createsuperuser

# run the dev server
python manage.py runserver
```

Test endpoints with curl / Postman / frontend:

Create Behaviour (example):

bash
Copy code
```
curl -X POST http://127.0.0.1:8000/api/behaviour/ \
  -H "Content-Type: application/json" \
  -d '{"student": 1, "category": 4, "custom_name": "Disruptive talk"}'
```
Create Attendance (example):

bash
Copy code
```
curl -X POST http://127.0.0.1:8000/api/attendance/ \
  -H "Content-Type: application/json" \
  -d '{"student":1,"status":"Absent","reason":"Sick"}'
```
If a student has two records with category: 4, the behaviour endpoint will:

set detention_triggered=True on the latest behaviour,

produce a voice file under backend/voice_files/alert_*.mp3,

return JSON like:

json
Copy code
```
{
  "triggered": true,
  "message": "Ali Khan has received two 'Poor' marks and is flagged for detention.",
  "voice_file": "backend/voice_files/alert_....mp3",
  "behaviour_id": 5
}
```
You can then have your React frontend load and play that MP3.

Notes & safety
The voice files are stored locally in voice_files/. In production you'd serve them as static files or upload them to storage (S3).

This system only triggers digital alerts (status flags and saved MP3s). It does not operate any real-world alarms or hardware (this is intentional and required for safety).

Change SECRET_KEY in settings.py before any deployment and set DEBUG = False for production.

For production DB use PostgreSQL (update DATABASES accordingly).