Funcionamiento del Sistema: Descripción Detallada

El sistema es una plataforma educativa dinámica que monitorea la atención de estudiantes mediante visión por computadora, calcula notas combinadas (40% atención + 60% académica), y genera rutas de aprendizaje adaptativas con retroalimentación personalizada. Los cursos se estructuran en **Curso → Niveles → Lecciones → Recursos/Actividades**, con un examen final por nivel. La IA (Claude) analiza atención y rendimiento para sugerir refuerzos, que el docente aprueba antes de enviar al estudiante. Todo se registra para seguimiento longitudinal, exportable en CSV/PDF.

Arquitectura General

- **Backend**: Django, Django REST Framework, PostgreSQL.
- **Visión por Computadora**: MediaPipe (FaceMesh, Iris, Holistic), OpenCV, WebRTC.
- **IA**: Claude API (Sonnet/Haiku) para rutas y retroalimentación.
- **Frontend**: React/NEXT.JS, Django Templates.
- Patrones de Atención:

  - Desviación sostenida de mirada (distracción, MediaPipe Iris).

  - Cierre prolongado de ojos (somnolencia, OpenCV).

  - Cabeza fuera de marco/inclinada (ausencia, MediaPipe FaceMesh).

  - Tiempo de permanencia frente a cámara (participación, MediaPipe Holistic).
- **Umbrales de Atención**: Alto (≥80%), Medio (50%–79%), Bajo (<50%), ajustables.
- **Exportación**: Reportes en CSV/PDF (jsPDF/XLSX).
- **Integración**: API genérica para LMS (Moodle, Canvas).

Roles y Funcionalidades

1. Administrador

Propósito: Gestiona la plataforma, usuarios, cursos y configuraciones globales.

Funcionalidades:

1. Gestión de Usuarios:

- Crea, edita, elimina cuentas de docentes y estudiantes.

- Asigna roles (Admin, Docente, Estudiante).

- Configura permisos (ej. docentes suben recursos, estudiantes acceden a cursos).

- Interfaz: Panel admin (Django Admin personalizado).

- Lista de usuarios con filtros (rol, estado).

- Formulario para crear/editar usuarios (nombre, email, rol).

- Implementación: Modelo Usuario (extiende AbstractUser):

python # usuarios/models.py from django.contrib.auth.models import AbstractUser class Usuario(AbstractUser): ROLES = [("admin", "Administrador"), ("docente", "Docente"), ("estudiante", "Estudiante")] rol = models.CharField(max_length=20, choices=ROLES, default="estudiante")

2. Gestión de Cursos:

- Crea cursos base (ej. “Cálculo I”) y asigna docentes.

- Configura umbrales institucionales (ej. nota mínima 70%, atención <50% = Bajo).

- Interfaz: Formulario para cursos (nombre, descripción, docente).

- Implementación: Modelo Curso:

python # cursos/models.py class Curso(models.Model): nombre = models.CharField(max_length=100) descripcion = models.TextField() docente = models.ForeignKey(Usuario, on_delete=models.CASCADE, limit_choices_to={"rol": "docente"}) umbral_nota = models.FloatField(default=70.0) # Umbral institucional

3. Configuraciones Globales:

- Ajusta umbrales de atención (Alto ≥80%, Medio 50%–79%, Bajo <50%) en settings.py:

python UMBRALES_ATENCION = {"ALTO": 80, "MEDIO": 50, "BAJO": 0} MAX_UPLOAD_SIZE = 104857600 # 100 MB

- Configura integración con LMS (API endpoints para exportar datos).

- Activa/desactiva módulos (ej. Claude API, monitoreo visual).

- Interfaz: Panel de configuración con formularios.

4. Reportes Generales:

- Genera reportes de uso (número de usuarios, cursos activos, métricas de atención promedio).

- Exporta en CSV/PDF.

- Interfaz: Dashboard con gráficos (Chart.js) y botón de exportación.

Flujo de Trabajo:

- Admin crea un curso ("Matemáticas 101"), asigna un docente, y configura umbral de nota (70%).
- Registra estudiantes y docentes en el sistema.
- Ajusta umbrales de atención si la institución lo requiere (ej. Alto ≥85%).
- Revisa reportes globales para auditorías (ISO 21001:2018).

2. Docente

Propósito: Gestiona cursos, sube recursos, supervisa estudiantes, aprueba retroalimentación IA, y recibe alertas.

Funcionalidades:

1. Gestión de Cursos:

- Configura niveles (ej. "Nivel 1: Álgebra Básica") y fases (ej. "Fase 1: Ecuaciones").

- Sube recursos/actividades:

- Formatos soportados:

- Videos (MP4, WebM, visor HTML5 con WebRTC).

- PDFs/docs (visor react-pdf para monitoreo).

- Quizzes/formularios (creados en plataforma).

- Simuladores (iframes/código personalizado).

- Archivos (docx, jpg, sin monitoreo).

- Límite: 100 MB por archivo.

- Define si la actividad es evaluable (nota académica) o no.

- Interfaz: Panel docente con formulario para subir recursos:

- Campos: Nombre, tipo (video, quiz, etc.), archivo, evaluable (sí/no).

- Advertencia si el recurso no permite monitoreo (ej. “Archivos docx no generan métricas de atención”).

- Implementación:

```python
# cursos/models.py
from django.core.validators import FileExtensionValidator
from django.db import models
class Nivel(models.Model):
    curso = models.ForeignKey(Curso, on_delete=models.CASCADE)
    nombre = models.CharField(max_length=100)
    orden = models.IntegerField()  # Para linealidad
class Fase(models.Model):
    nivel = models.ForeignKey(Nivel, on_delete=models.CASCADE)
    nombre = models.CharField(max_length=100)
class Recurso(models.Model):
    TIPO = [("video", "Video"), ("quiz", "Quiz"), ("pdf", "PDF"), ("simulador", "Simulador"), ("archivo", "Archivo")]
    fase = models.ForeignKey(Fase, on_delete=models.CASCADE)
    nombre = models.CharField(max_length=100)
    tipo = models.CharField(max_length=20, choices=TIPO)
    archivo = models.FileField(upload_to="recursos/", validators=[FileExtensionValidator(allowed_extensions=["mp4", "webm", "pdf", "docx", "jpg", "png"])])
    permite_monitoreo = models.BooleanField(default=True)
    es_evaluable = models.BooleanField(default=False)
```

2. Monitoreo de Estudiantes:

- Recibe reportes por estudiante:

- Tabla: Nombre, estado (Listo 🟢, Observación 🟠, Refuerzo 🔴), nota académica, score de atención, recomendación IA.

- Gráficos: Atención por fase (barras, Chart.js), evolución longitudinal.

- Secciones:

- Rendimiento académico (notas por actividad).

- Atención visual (score, patrones: mirada desviada, etc.).

- Recomendaciones activas (editables).

- Historial reciente (progreso por fase).

- Interfaz: Dashboard docente con filtros (estudiante, fase, curso).

3. Aprobación de Retroalimentación IA:

- IA (Claude) genera sugerencias personalizadas para estudiantes en "Observación" o "Refuerzo":

- Tipos: Refuerzo de contenido, técnica de estudio, revisar foco, repetir actividad.

- Ejemplo: "Juan, tu atención en 'Ecuaciones' fue baja (<50%). Revisa este video."

- Docente revisa sugerencias en panel:

- Aprueba (envía al estudiante).

- Edita (personaliza mensaje).

- Descarta (no se envía).

- Interfaz: Tabla de sugerencias con botones "Aprobar/Editar/Descartar".

- Implementación:

python # recomendaciones/models.py class RecomendacionIA(models.Model): estudiante = models.ForeignKey(Usuario, on_delete=models.CASCADE) fase = models.ForeignKey(Fase, on_delete=models.CASCADE) tipo = models.CharField(max_length=50, choices=[("contenido", "Refuerzo de contenido"), ("tecnica", "Técnica de estudio"), ("foco", "Revisar foco"), ("repetir", "Repetir actividad")]) mensaje = models.TextField() # Ej. "Revisa video X" estado = models.CharField(max_length=20, choices=[("pendiente", "Pendiente"), ("aprobada", "Aprobada"),

("descartada", "Descartada")]) docente_aprobo = models.BooleanField(default=False) fecha = models.DateTimeField(auto_now_add=True)

4. Alertas:

   - Notificaciones automáticas para estudiantes en “Refuerzo” (🔴) o con bajo progreso repetido.

   - Ejemplo: “Estudiante Juan lleva 2 fases con atención <50%.”

   - Interfaz: Sección de alertas en dashboard (prioridad alta para casos críticos).

5. Exportación de Reportes:

   - Descarga reportes por estudiante/curso (CSV/PDF).

   - Incluye: Notas, scores de atención, recomendaciones aplicadas, historial.

   - Implementación: Librerías jsPDF/XLSX en frontend.

Flujo de Trabajo:

- Docente crea nivel (“Álgebra Básica”) y fases (“Ecuaciones Lineales”).
- Sube video (MP4) y quiz interno, marca el quiz como evaluable.
- Revisa reportes: Estudiante Juan en “Refuerzo” (atención 40%, nota 60%).
- Aprueba sugerencia IA: “Revisa video ‘Ecuaciones Básicas’.”
- Recibe alerta si Juan no mejora tras refuerzo.
- Descarga reporte de curso en CSV.

3. Estudiante

Propósito: Interactúa con el curso, recibe retroalimentación personalizada, y sigue una ruta de aprendizaje adaptativa.

Funcionalidades:

1. Acceso a Cursos:

   - Visualiza cursos inscritos (ej. “Matemáticas 101”).

   - Navega niveles/fases linealmente (Nivel 1 → Nivel 2).

   - Accede a recursos/actividades (videos, PDFs en visor, quizzes, simuladores).

- Interfaz: Dashboard estudiante con barra de progreso (porcentaje completado por fase).

2. Monitoreo de Atención:

   - Durante interacción con recursos (videos, PDFs en visor, quizzes, simuladores):

     - WebRTC activa cámara frontal.

     - MediaPipe/OpenCV detecta:

       - Desviación de mirada (Iris).

       - Cierre prolongado de ojos (OpenCV).

       - Cabeza fuera de marco/inclinada (FaceMesh).

       - Tiempo de permanencia (Holistic).

     - Calcula score de atención (0-100%) por recurso:

     python # atencion/models.py class AtencionVisual(models.Model): estudiante = models.ForeignKey(Usuario, on_delete=models.CASCADE) recurso = models.ForeignKey(Recurso, on_delete=models.CASCADE) score_atencion = models.FloatField() # 0-100 patrones = models.JSONField() # Ej. {"desviacion_gaze": 0.4, "cierre_ojos": 0.3} fecha = models.DateTimeField(auto_now_add=True)

     - Score: Alto (≥80%), Medio (50%–79%), Bajo (<50%).

3. Rutas de Aprendizaje:

   - Estructura: Lineal (Nivel 1 → Nivel 2), con refuerzos personalizados.

   - Análisis IA:

     - Por fase, IA (Claude) evalúa:

       - Atención: Score promedio de recursos.

       - Notas: Promedio de actividades evaluables (vs. umbral institucional).

     - Genera estado: Listo (🟢), Observación (🟠), Refuerzo (🔴).

   - Refuerzos:

     - Si “Observación” o “Refuerzo”, IA sugiere recursos extra (video, quiz, etc.).

- Docente aprueba → Estudiante ve recurso en su ruta.

- Interfaz: Sección “Ruta de Aprendizaje” con:

- Progreso (gráfico de avance).

- Recursos obligatorios y complementarios.

4. Retroalimentación Personalizada:

   - Recibe mensajes extensos y detallados (aprobados por docente):

   - Ejemplo: “Juan, tu atención en ‘Ecuaciones’ fue baja (40%). Resuelve este quiz y revisa el video ‘Ecuaciones Básicas’ para mejorar.”

   - No ve métricas técnicas (scores, patrones), solo notas y mensajes.

   - Interfaz: Sección “Retroalimentación” con mensajes y enlaces a recursos.

5. Evaluación Final:

   - Desbloqueada tras completar actividades obligatorias.

   - Nota combinada: (0.4 × Promedio atención) + (0.6 × Promedio notas académicas).

   - Umbral de aprobación: Definido por institución (ej. ≥70%).

   - Interfaz: Botón “Rendir Examen” en dashboard.

6. Historial:

   - Consulta su progreso (notas, fases completadas, refuerzos aplicados).

   - No ve métricas de atención detalladas.

   - Interfaz: Sección “Historial” con lista de fases y notas.

Flujo de Trabajo:

- Estudiante accede a “Nivel 1: Álgebra Básica”, Fase 1.
- Ve video “Ecuaciones” (cámara activa, MediaPipe calcula atención: 40%).
- Resuelve quiz (nota: 60%).
- IA detecta “Refuerzo” (🔴), sugiere video extra.
- Docente aprueba: “Juan, revisa ‘Ecuaciones Básicas’.”
- Estudiante ve mensaje y video extra en su ruta.
- Completa actividades, desbloquea examen final.

- Nota combinada: (0.4 × 40) + (0.6 × 60) = 52%. Si no aprueba, recibe más refuerzos.

Flujo General del Sistema

1. Inicio:

   - Admin crea curso, asigna docente, registra estudiantes.

   - Docente configura niveles/fases, sube recursos (videos, quizzes, PDFs).

2. Interacción:

   - Estudiante accede a fase, interactúa con recursos.

   - MediaPipe/OpenCV calcula score de atención (0-100%) por recurso.

   - Actividades evaluables generan nota académica.

3. Análisis IA:

   - Claude analiza:

   - Atención: Alto (≥80%), Medio (50%–79%), Bajo (<50%).

   - Notas: Vs. umbral institucional (ej. 70%).

   - Genera estado (Listo, Observación, Refuerzo) y sugerencia:

   - Tipos: Refuerzo de contenido, técnica de estudio, revisar foco, repetir actividad.

   - Almacena en RecomendacionIA (estado “Pendiente”).

4. Retroalimentación:

   - Docente revisa sugerencia en panel:

   - Aprueba: Estado → “Aprobada”, mensaje enviado.

   - Edita: Personaliza mensaje.

   - Descarta: No se envía.

   - Estudiante ve mensaje personalizado (extenso, detallado) y recursos extra.

5. Ruta de Aprendizaje:

   - Lineal: Estudiante avanza por fases/niveles.

   - Refuerzos: Recursos extra añadidos si “Observación”/“Refuerzo”.

   - Almacenado en HistorialEstudiante:

python # recomendaciones/models.py from django.db import models class HistorialEstudiante(models.Model): estudiante = models.ForeignKey(Usuario, on_delete=models.CASCADE) curso = models.ForeignKey(Curso, on_delete=models.CASCADE) nivel = models.ForeignKey(Nivel, on_delete=models.CASCADE) fase = models.ForeignKey(Fase, on_delete=models.CASCADE) actividad = models.ForeignKey(Recurso, on_delete=models.CASCADE, null=True) score_atencion = models.FloatField(null=True) nota_academica = models.FloatField(null=True) recomendacion = models.ForeignKey(RecomendacionIA, on_delete=models.SET_NULL, null=True) estado = models.CharField(max_length=20, choices=[("Listo", "Listo"), ("Observacion", "Observación"), ("Refuerzo", "Refuerzo")]) fecha = models.DateTimeField(auto_now_add=True)

6. Evaluación Final:

   - Desbloqueada tras completar actividades.

   - Nota combinada: (0.4 × Atención) + (0.6 × Académica).

   - Refuerzos no afectan nota, pero mejoran preparación.

7. Reportes:

   - Docente: Reportes detallados (tabla, gráficos, historial).

   - Admin: Reportes globales.

   - Exportables en CSV/PDF.

Ejemplo Práctico

Curso: “Matemáticas 101”.

Nivel: “Álgebra Básica”.

Fase: “Ecuaciones Lineales”.

Recursos: Video “Ecuaciones” (MP4), Quiz interno.

1. Admin:

   - Crea curso, asigna docente Ana, registra estudiante Juan.

   - Configura umbral institucional: 70%.

2. Docente (Ana):

   - Sube video y quiz (evaluable).

- Recibe reporte: Juan (atención 40%, nota 60%, estado “Refuerzo”).

- Aprueba sugerencia IA: “Juan, revisa ‘Ecuaciones Básicas’ y resuelve quiz extra.”

3. Estudiante (Juan):

- Ve video (cámara activa, MediaPipe detecta atención baja).

- Resuelve quiz (nota 60%).

- Recibe mensaje: “Tu atención fue baja. Revisa este video y quiz extra.”

- Completa refuerzo, desbloquea examen final.

- Nota combinada: (0.4 × 40) + (0.6 × 60) = 52% (no aprueba, recibe más refuerzos).

Implementación Técnica

1. Backend:

- Modelos:

- Usuario: Roles (Admin, Docente, Estudiante).

- Curso, Nivel, Fase, Recurso: Estructura del curso.

- AtencionVisual: Scores de atención por recurso.

- RecomendacionIA: Sugerencias personalizadas.

- HistorialEstudiante: Registro longitudinal.

- APIs (Django REST Framework):

- /api/cursos/: Gestión de cursos.

- /api/recursos/: Subida de recursos.

- /api/atencion/: Scores de atención.

- /api/recomendaciones/: Generar/aprobar sugerencias.

- Lógica:

python from django.db.models import Avg def calcular_nota_combinada(estudiante, fase): atencion = AtencionVisual.objects.filter(estudiante=estudiante, fase=fase).aggregate(Avg("score_atencion"))["score_atencion__avg"] or 0 notas = Recurso.objects.filter(fase=fase, es_evaluable=True).aggregate(Avg("nota"))["nota__avg"] or 0 return (0.4 * atencion) + (0.6 * notas)

2. Visión por Computadora:

   - MediaPipe/OpenCV calcula score de atención:

   python def calcular_score_atencion(frame): weights = {"desviacion_gaze": 0.4, "cierre_ojos": 0.3, "cabeza_ausente": 0.2, "tiempo_permanencia": 0.1} score = 100 - sum(weights[p] * detectar_patron(p, frame) for p in weights) return max(0, min(100, score))

   - WebRTC captura video en tiempo real.

3. IA (Claude):

   - Genera sugerencias basadas en atención y notas:

   python def generar_sugerencia_claude(atencion, notas, fase): prompt = f"Estudiante con atención {atencion}% y nota {notas} en fase {fase.nombre}. Sugiere acción." response = claude_api.call(prompt) return {"tipo": response["tipo"], "mensaje": response["mensaje"]}

4. Frontend:

   - Estudiante:

     - Progreso (Chart.js).

     - Retroalimentación (mensajes personalizados).

     - Recursos extra (enlaces).

   - Docente:

     - Tabla de estudiantes (estado, notas, atención).

     - Gráficos (Chart.js).

     - Botones para aprobar/editar sugerencias.

   - Admin: Panel de gestión (Django Admin).

Confirmación con Requisitos

- **Rutas de aprendizaje**: Lineales, con refuerzos personalizados (IA, aprobados por docente).
- **Retroalimentación**: Extensa, detallada, personalizada, enviada tras aprobación docente.
- **Umbrales**: Alto (≥80%), Medio (50%–79%), Bajo (<50%), ajustables.

- **Formatos recursos**: Videos, PDFs (visor), quizzes, simuladores, archivos (100 MB).
- **Roles**: Admin (gestión), Docente (supervisión), Estudiante (aprendizaje).
- **Herramientas**: Django, PostgreSQL, MediaPipe, OpenCV, Claude, React/NEXT.JS.
- **Cumplimiento**: ISO 21001:2018, integrable con LMS.

Pregunta final: ¿Es este el funcionamiento que buscas? ¿Falta algún detalle o necesitas ajustes (ej. más funcionalidades para admin, notificaciones push)?

Siguientes Pasos Inmediatos

1. Confirmar Funcionamiento:

   - ¿El flujo descrito cubre tu visión? Si no, señala ajustes.

2. Continuar Etapa 1:

   - Completar configuración Django (modelos iniciales: Usuario, Curso, Nivel, Fase, Recurso).

   - Probar script MediaPipe (detección facial).

   - Crear repositorio GitHub.

3. Entregables:

   - Código: Proyecto Django, apps, models.py, script MediaPipe.

   - Opcional: Diagrama de flujo (si lo pides).

   - Pregunta: ¿Sigo con el código (modelos iniciales) o genero diagrama primero?

Código Inicial (Modelos):

```
# usuarios/models.py

from django.contrib.auth.models import AbstractUser

from django.db import models


class Usuario(AbstractUser):

  ROLES = [("admin", "Administrador"), ("docente", "Docente"), ("estudiante", "Estudiante")]
```

```
    rol = models.CharField(max_length=20, choices=ROLES, default="estudiante")


# cursos/models.py
from django.core.validators import FileExtensionValidator
class Curso(models.Model):
    nombre = models.CharField(max_length=100)
    descripcion = models.TextField()
    docente = models.ForeignKey(Usuario, on_delete=models.CASCADE, limit_choices_to={"rol": "docente"})
    umbral_nota = models.FloatField(default=70.0)

class Nivel(models.Model):
    curso = models.ForeignKey(Curso, on_delete=models.CASCADE)
    nombre = models.CharField(max_length=100)
    orden = models.IntegerField()

class Fase(models.Model):
    nivel = models.ForeignKey(Nivel, on_delete=models.CASCADE)
    nombre = models.CharField(max_length=100)

class Recurso(models.Model):
    TIPO = [("video", "Video"), ("quiz", "Quiz"), ("pdf", "PDF"), ("simulador", "Simulador"), ("archivo", "Archivo")]
    fase = models.ForeignKey(Fase, on_delete=models.CASCADE)
    nombre = models.CharField(max_length=100)
    tipo = models.CharField(max_length=20, choices=TIPO)
    archivo = models.FileField(upload_to="recursos/", validators=[FileExtensionValidator(allowed_extensions=["mp4", "webm", "pdf", "docx", "jpg", "png"])])
```

```
permite_monitoreo = models.BooleanField(default=True)

es_evaluable = models.BooleanField(default=False)
```