💼 Kipps.AI Internship Assignment

Title: Post-Conversation Analysis

Objective:
Build an automated Django application that performs post-conversation analysis on chat messages (AI + human) and stores the results in a database.

🧠 Part 1 — Post Conversation Analysis

You are required to perform **post-conversation analysis** on chat messages between an **AI agent** and a **human user**.

Each conversation will consist of a list of messages in this format:

```
[
  {"sender": "user", "message": "Hi, I need help with my order."},
  {"sender": "ai", "message": "Sure, can you please share your order ID?"},
  {"sender": "user", "message": "It's 12345."},
  {"sender": "ai", "message": "Thanks! Your order has been shipped and will arrive tomorrow."}
]
```

Your analysis should include the following **parameters** (minimum 10 from the list below):

Category	Parameter	Description
Conversation Quality	Clarity	Was the response clear and easy to understand?
	Relevance	Did the AI stay on topic?
	Accuracy	Were responses factually correct?
	Completeness	Did the AI provide a complete answer?

Interaction	Sentiment	Determine user sentiment (positive, neutral, negative).
	Empathy	Did AI respond empathetically when needed?
	Response Time	Average time difference between messages (mock data acceptable).
Resolution	Resolution Rate	Was the issue resolved in the conversation?
	Escalation Need	Should this chat be escalated to a human?
AI Ops	Fallback Frequency	Count how many times AI used fallback (e.g., “I don’t know”).
Overall	User Satisfaction Score	Compute a final score (e.g., average of key parameters).

⚙️ Part 2 — Django Application

Create a **Django REST Framework** application that:

1. Accepts JSON chat input (via API or file upload).
2. Performs post-conversation analysis using the parameters above.
3. Stores the results in a PostgreSQL/SQLite database.

Models (example)

```
class Conversation(models.Model):
    title = models.CharField(max_length=255)
    created_at = models.DateTimeField(auto_now_add=True)

class Message(models.Model):
    conversation = models.ForeignKey(Conversation,
on_delete=models.CASCADE, related_name="messages")
    sender = models.CharField(max_length=20)  # "user" or "ai"
    text = models.TextField()

class ConversationAnalysis(models.Model):
```

```
    conversation = models.OneToOneField(Conversation,
on_delete=models.CASCADE, related_name="analysis")
    clarity_score = models.FloatField()
    relevance_score = models.FloatField()
    sentiment = models.CharField(max_length=20)
    empathy_score = models.FloatField()
    resolution = models.BooleanField()
    overall_score = models.FloatField()
    created_at = models.DateTimeField(auto_now_add=True)
```

Endpoints (example)

Endpoint	Method	Description
`/api/conversations/`	`POST`	Upload a chat JSON
`/api/analyse/`	`POST`	Trigger analysis on a conversation
`/api/reports/`	`GET`	List all conversation analysis results

🕛 Part 3 — Cron Job Automation

Add a **cron job or Celery periodic task** that:

- Runs **once a day** (e.g., at 12 AM).
- Automatically performs analysis on **all new chat conversations** and updates the database.
- Use **Django-crontab** or **Celery Beat**.

🧩 Part 4 — Deliverables

1. **Working Django app** with:
    - API endpoints
    - Automated daily cron job
    - Database integration

2. Include all dependencies in `requirements.txt`

3. **README.md** with:
    - Set up and run instructions
    - Cron job setup steps
    - API documentation with examples
    - Short Loom video (optional) explaining your architecture

Public GitHub Repository URL

Example:

`https://github.com/<your-username>/post-conversation-analysis`

✅ Submission Guidelines

- Deadline: **72 hours from assignment receipt**
- Deliverables:
    - GitHub repo link