=== PAGE 1 ===
RAG MICROSERVICES â\200\224 THE ULTIMATE BEGINNERâ\200\231S SELF-TEACHING GUIDE
==========

Welcome, future elite developer.

This document is your companion â\200\224 not your crutch.

You will not be handed code.
You will be handed **process**.
You will be taught **how to search**.
You will be guided to **think, try, fail, fix, and THEN understand**.

Each step is broken into substeps.
Each substep tells you:
â\206\222 What to DO
â\206\222 Where to SEARCH
â\206\222 What to THINK
â\206\222 What ERROR you might see
â\206\222 How to DEBUG
â\206\222 And ONLY THEN â\200\224 what CODE you should end up with â\200\224 and what it do
es.

This document will become your RAG systemâ\200\231s first knowledge base.

Letâ\200\231s begin.

=== PAGE 2 ===
STEP 1: CREATE PROJECT FOLDER â\200\224 SUBSTEP BY SUBSTEP
==========

ð\237\224¹ SUBSTEP 1.1: Open your terminal

â\206\222 How?
   - Mac: Press Cmd+Space â\206\222 type â\200\234Terminalâ\200\235 â\206\222 Enter.
   - Linux: Ctrl+Alt+T.
   - Windows: Search â\200\234Command Promptâ\200\235 or â\200\234PowerShellâ\200\235.

ð\237\222¡ GOAL: You need a command-line interface to create folders.

â\200\224

ð\237\224¹ SUBSTEP 1.2: Navigate to your home directory (optional but clean)

â\206\222 Type: cd ~
â\206\222 Press Enter.

ð\237\222¡ WHY? So you know exactly where your project lives.

â\234\205 WHAT THIS COMMAND DOES:
Changes your current location to your home folder (e.g., /home/yourname or /Users/yourname)
.

â\200\224

ð\237\224¹ SUBSTEP 1.3: Create the main project folder

â\206\222 Type: mkdir -p rag-microservices/document-ingestor
â\206\222 Press Enter.

ð\237\222¡ IF YOU FORGET WHAT mkdir DOES:
â\206\222 Open browser â\206\222 search: â\200\234linux mkdir commandâ\200\235
â\206\222 Youâ\200\231ll learn: mkdir = â\200\234make directoryâ\200\235

â\234\205 WHAT THIS COMMAND DOES:
Creates a folder called â\200\234rag-microservicesâ\200\235, and inside it, â\200\234docume
nt-ingestorâ\200\235. The -p flag means â\200\234create parents if neededâ\200\235.

â\200\224

ð\237\224¹ SUBSTEP 1.4: Navigate into the folder

â\206\222 Type: cd rag-microservices/document-ingestor
â\206\222 Press Enter.

ð\237\222¡ VERIFY: Type pwd â\206\222 should show full path to this folder.

â\234\205 WHAT THIS COMMAND DOES:
Changes your current working directory â\200\224 so when you create files, they go here.

â\200\224

ð\237\222¡ SAMPLE Q&A:

Q: What if Iâ\200\231m on Windows and mkdir doesnâ\200\231t work?
A: It does â\200\224 but use backslashes: mkdir rag-microservices\document-ingestor

Q: Can I use a different folder name?
A: Yes â\200\224 but stick to this for consistency with this guide.

Q: What does â\200\234cdâ\200\235 mean?
A: â\200\234Change Directoryâ\200\235 â\200\224 it moves you between folders.

â\200\224

â\234\205 FINAL STRUCTURE AFTER THIS STEP:
~/rag-microservices/document-ingestor/ â\206\220 you are here

=== PAGE 3 ===
STEP 2: CREATE PYTHON FILE â\200\224 SUBSTEP BY SUBSTEP
==========

ð\237\224¹ SUBSTEP 2.1: Create the file ingest_pdf.py

â\206\222 Type: touch ingest_pdf.py
â\206\222 Press Enter.

ð\237\222¡ IF YOU DONâ\200\231T KNOW WHAT touch DOES:
â\206\222 Search: â\200\234linux touch commandâ\200\235
â\206\222 Learn: It creates an empty file.

â\234\205 WHAT THIS COMMAND DOES:
Creates a new, empty file named â\200\234ingest_pdf.pyâ\200\235 in your current folder.

â\200\224

ð\237\224¹ SUBSTEP 2.2: Verify it exists

â\206\222 Type: ls -l
â\206\222 Press Enter.

ð\237\222¡ EXPECT TO SEE: ingest_pdf.py in the list.

â\234\205 WHAT THIS COMMAND DOES:
Lists all files in current folder â\200\224 with details (permissions, size, etc.).

â\200\224

ð\237\224¹ SUBSTEP 2.3: Open it in your editor

â\206\222 If you have VS Code: code ingest_pdf.py
â\206\222 If you have nano: nano ingest_pdf.py
â\206\222 If on Windows: Right-click â\206\222 Open with Notepad

ð\237\222¡ IF YOU DONâ\200\231T HAVE VS CODE:
â\206\222 Search: â\200\234install vscode linux/mac/windowsâ\200\235 â\206\222 follow offic

ial guide.

â\234\205 WHAT THIS STEP DOES:
Opens the file so you can write code inside it.

â\200\224

ð\237\222¡ SAMPLE Q&A:

Q: What if touch doesnâ\200\231t work on Windows?
A: Use: type nul > ingest_pdf.py OR create via File Explorer â\206\222 New â\206\222 Text Document â\206\222 rename to .py

Q: Should I write code now?
A: No â\200\224 not yet. Just create and open the file.

Q: Why .py extension?
A: It tells your system this is a Python file.

â\200\224

â\234\205 YOU NOW HAVE: An empty Python file â\200\224 ready for your first lines of code.

=== PAGE 4 ===
STEP 3: SEARCH FOR PDF LOADER â\200\224 SUBSTEP BY SUBSTEP
==

ð\237\224¹ SUBSTEP 3.1: Open your web browser

â\206\222 Chrome, Firefox, Edge â\200\224 any is fine.

â\200\224

ð\237\224¹ SUBSTEP 3.2: Search for the tool

â\206\222 In search bar, type exactly: langchain load pdf example
â\206\222 Press Enter.

ð\237\222¡ WHY THIS SEARCH?
Because you want to load a PDF â\200\224 and LangChain is the most popular tool for this in RAG systems.

â\200\224

ð\237\224¹ SUBSTEP 3.3: Click the official result

â\206\222 Look for: â\200\234PDF â\200\224 ð\237¦\234ï¸\217ð\237\224\227 LangChainâ\200\235
â\206\222 URL should be: https://python.langchain.com/docs/modules/data_connection/document_loaders/pdf

ð\237\222¡ IF YOU SEE BLOGS OR VIDEOS:
â\206\222 Skip them. Only trust official docs for now.

â\234\205 WHAT YOUâ\200\231RE LOOKING FOR:
- The name of the loader (starts with â\200\234PyPDFâ\200¦â\200\235)
- What package to install (hint: pypdf)
- Example code using .load()

â\200\224

ð\237\224¹ SUBSTEP 3.4: Read â\200\224 donâ\200\231t copy

â\206\222 Scan the page.
â\206\222 Note down:
   - Import statement
   - Class name
   - Method to load
   - Output structure

ð\237\222¡ GOAL: Understand â\200\224 not copy.

â\200\224

ð\237\222¡ SAMPLE Q&A:

Q: What is LangChain?
A: A framework for building apps with LLMs â\200\224 like RAG.

Q: Why not use PyPDF2 directly?
A: LangChain gives you a standard interface â\200\224 easier to swap tools later.

Q: What if the page looks too complex?
A: Focus only on the PyPDFLoader section. Ignore the rest for now.

â\200\224

â\234\205 YOU NOW KNOW: You need PyPDFLoader from langchain_community, and to install pypdf
.

=== PAGE 5 ===
STEP 4: CREATE VIRTUAL ENVIRONMENT â\200\224 SUBSTEP BY SUBSTEP
==========

ð\237\224¹ SUBSTEP 4.1: Go back to project root

â\206\222 Type: cd ../..
â\206\222 Press Enter.

ð\237\222¡ WHY? Because virtual environments should live at project root â\200\224 so all s
ervices can share it (for now).

â\234\205 WHAT THIS COMMAND DOES:
Moves you up two levels â\200\224 from document-ingestor â\206\222 rag-microservices â\206
\222 ~ (or wherever rag-microservices lives).

â\200\224

ð\237\224¹ SUBSTEP 4.2: Create the virtual environment

â\206\222 Type: python -m venv venv
â\206\222 Press Enter.

ð\237\222¡ IF YOU GET ERROR:
â\206\222 Search: â\200\234python -m venv not workingâ\200\235
â\206\222 You might need to install python3-venv (Linux) or use python3 instead of python.

â\234\205 WHAT THIS COMMAND DOES:
Creates a folder called â\200\234venvâ\200\235 â\200\224 which contains an isolated Python
environment.

â\200\224

ð\237\224¹ SUBSTEP 4.3: Activate it

â\206\222 Mac/Linux: source venv/bin/activate
â\206\222 Windows: venv\Scripts\activate

ð\237\222¡ VERIFY: You should see (venv) at the start of your terminal line.

â\234\205 WHAT THIS COMMAND DOES:
Activates the isolated environment â\200\224 so any packages you install stay here.

â\200\224

ð\237\224¹ SUBSTEP 4.4: Install required packages

â\206\222 Type: pip install pypdf langchain-community
â\206\222 Press Enter.

ð\237\222¡ IF SLOW: Thatâ\200\231s normal â\200\224 itâ\200\231s downloading and installing .

â\234\205 WHAT THIS COMMAND DOES:
Installs the libraries you need to load PDFs.

â\200\224

ð\237\224¹ SUBSTEP 4.5: Save your dependencies

â\206\222 Type: pip freeze > requirements.txt
â\206\222 Press Enter.

â\234\205 WHAT THIS COMMAND DOES:
Creates a file listing every installed package + version â\200\224 so you (or anyone) can r ecreate this environment.

â\200\224

ð\237\222¡ SAMPLE Q&A:

Q: What is a virtual environment?
A: A sandbox â\200\224 keeps your projectâ\200\231s packages separate from others.

Q: Why not install globally?
A: Avoids version conflicts. Professional projects always use venv.

Q: What if pip freeze is empty?
A: You forgot to activate venv. Run source venv/bin/activate first.

â\200\224

â\234\205 YOU NOW HAVE: An isolated, reproducible Python environment â\200\224 ready for co ding.

=== PAGE 6 ===
STEP 5: PREPARE TEST PDF â\200\224 SUBSTEP BY SUBSTEP
==========

ð\237\224¹ SUBSTEP 5.1: Create sample.txt

â\206\222 You are reading it now â\200\224 this entire guide is your sample.txt.

â\206\222 Save this file as â\200\234sample.txtâ\200\235 in your document-ingestor folder.

â\200\224

ð\237\224¹ SUBSTEP 5.2: Convert to PDF

OPTION A (Linux/Mac with enscript):
â\206\222 Type: enscript -p - sample.txt | ps2pdf - sample.pdf

OPTION B (Google Docs):
â\206\222 Open sample.txt â\206\222 copy all â\206\222 paste into new Google Doc â\206\222 File â\206\222 Download â\206\222 PDF â\206\222 save as sample.pdf

OPTION C (Word):
â\206\222 Paste into Word â\206\222 Save As â\206\222 PDF â\206\222 sample.pdf

â\200\224

ð\237\224¹ SUBSTEP 5.3: Move it to the right folder

â\206\222 Make sure sample.pdf is in document-ingestor â\200\224 same folder as ingest_pdf. py.

â\206\222 Verify: ls â\206\222 should show both files.

â\200\224

ð\237\222¡ SAMPLE Q&A:

Q: What if my PDF has no text?
A: Youâ\200\231ll see len(page.page_content) = 0 â\200\224 then you know itâ\200\231s scann
ed â\200\224 use this text-based one.

Q: Can I use my own PDF?
A: Yes â\200\224 but if itâ\200\231s scanned, switch to this one for learning.

Q: Why not start with a real business PDF?
A: Because youâ\200\231re learning â\200\224 start simple, then scale.

â\200\224

â\234\205 YOU NOW HAVE: A guaranteed text-based PDF â\200\224 perfect for testing.

=== PAGE 7 ===
STEP 6: LOAD AND PRINT PDF â\200\224 SUBSTEP BY SUBSTEP
==========

ð\237\224¹ SUBSTEP 6.1: Open ingest_pdf.py

â\206\222 Terminal: code ingest_pdf.py (or nano, etc.)

â\200\224

ð\237\224¹ SUBSTEP 6.2: Write the import

â\206\222 Type: from langchain_community.document_loaders import

â\206\222 Try to remember the rest.

ð\237\222¡ IF STUCK: Go back to LangChain docs â\206\222 find the import.

â\234\205 WHAT YOU SHOULD WRITE:
from langchain_community.document_loaders import PyPDFLoader

â\234\205 WHAT IT DOES: Makes the PDF loader available in your script.

â\200\224

ð\237\224¹ SUBSTEP 6.3: Create the loader

â\206\222 Type: loader =

â\206\222 What goes next?

ð\237\222¡ HINT: You need to create a new PyPDFLoader â\200\224 and give it a filename.

â\234\205 WHAT YOU SHOULD WRITE:
loader = PyPDFLoader("sample.pdf")

â\234\205 WHAT IT DOES: Points the loader to your file â\200\224 doesnâ\200\231t load yet.

â\200\224

ð\237\224¹ SUBSTEP 6.4: Load the pages

â\206\222 Type: pages =

â\206\222 How to load?

ð\237\222¡ HINT: In docs, they use .load()

â\234\205 WHAT YOU SHOULD WRITE:
pages = loader.load()

â\234\205 WHAT IT DOES: Reads the PDF â\200\224 returns list of Document objects.

â\200\224

ð\237\224¹ SUBSTEP 6.5: Print metadata

â\206\222 Type: print(pages[0].metadata)

â\234\205 WHAT IT DOES: Shows info like page number, source, producer.

â\200\224

ð\237\224¹ SUBSTEP 6.6: Print content

â\206\222 Type: print(pages[0].page_content)

â\234\205 WHAT IT DOES: Shows actual text from first page.

â\200\224

ð\237\224¹ SUBSTEP 6.7: Print lengths (debug)

â\206\222 Type:
```
for i, page in enumerate(pages):
    print(f"Page {i+1} length: {len(page.page_content)} characters")
```

â\234\205 WHAT IT DOES: Helps you verify text was extracted â\200\224 if 0, PDF is scanned.

â\200\224

ð\237\224¹ SUBSTEP 6.8: Run it

â\206\222 Save file â\206\222 terminal â\206\222 python3 ingest_pdf.py

â\200\224

ð\237\222¡ SAMPLE Q&A:

Q: ModuleNotFoundError?
A: Activate venv â\206\222 pip install missing package.

Q: FileNotFoundError?
A: Is sample.pdf in the right folder? Run ls to check.

Q: Empty content?
A: Use text-based PDF â\200\224 not scanned.

â\200\224

â\234\205 FINAL CODE (but you typed it yourself!):
[See previous page for full code + line-by-line explanation]

=== PAGE 8 ===
STEP 7: SPLIT TEXT INTO CHUNKS â\200\224 SUBSTEP BY SUBSTEP (COMING NEXT)
==========

ð\237\224¹ SUBSTEP 7.1: Search for text splitter

â\206\222 Browser â\206\222 search: â\200\234langchain text splitter exampleâ\200\235

â\206\222 Click: https://python.langchain.com/docs/modules/data_connection/text_splitters/

â\200\224

ð\237\224¹ SUBSTEP 7.2: Find the right splitter

â\206\222 Look for: RecursiveCharacterTextSplitter

â\200\224

ð\237\224¹ SUBSTEP 7.3: Learn parameters

â\206\222 chunk_size = max characters per chunk (try 300)
â\206\222 chunk_overlap = chars to repeat (try 50)

â\200\224

ð\237\224¹ SUBSTEP 7.4: Import it

â\206\222 Type: from langchain.text_splitter import RecursiveCharacterTextSplitter

â\200\224

ð\237\224¹ SUBSTEP 7.5: Create splitter

â\206\222 Type: splitter = RecursiveCharacterTextSplitter(chunk_size=300, chunk_overlap=50)

â\200\224

ð\237\224¹ SUBSTEP 7.6: Split documents

â\206\222 Type: chunks = splitter.split_documents(pages)

â\200\224

ð\237\224¹ SUBSTEP 7.7: Print chunks

â\206\222 Type:
```
for i, chunk in enumerate(chunks[:3]):
    print(f"--- Chunk {i+1} ---\n{chunk.page_content}\n")
```

â\200\224

ð\237\222¡ SAMPLE Q&A:

Q: Why split text?
A: LLMs canâ\200\231t read long docs at once â\200\224 we break them into bite-sized pieces
.

Q: What is chunk_overlap?
A: Repeats end of previous chunk â\200\224 so context isnâ\200\231t lost at boundaries.

Q: Can I split by sentence?
A: Yes â\200\224 but RecursiveCharacter is best for beginners.

â\200\224

â\234\205 YOU WILL LEARN: How to prepare text for embedding â\200\224 next critical RAG ste
p.

=== PAGE 9 ===
STEP 8: GENERATE EMBEDDINGS â\200\224 SUBSTEP BY SUBSTEP (FUTURE)
==========

ð\237\224¹ SUBSTEP 8.1: Search for embedding model

â\206\222 Search: â\200\234sentence transformers exampleâ\200\235

â\206\222 Go to: www.sbert.net

â\200\224

ð\237\224¹ SUBSTEP 8.2: Install

â\206\222 pip install sentence-transformers

â\200\224

ð\237\224¹ SUBSTEP 8.3: Import model

â\206\222 from sentence_transformers import SentenceTransformer

â\200\224

ð\237\224¹ SUBSTEP 8.4: Load model

â\206\222 model = SentenceTransformer('all-MiniLM-L6-v2')

â\200\224

ð\237\224¹ SUBSTEP 8.5: Embed chunks

â\206\222 embeddings = model.encode([chunk.page_content for chunk in chunks])

â\200\224

ð\237\224¹ SUBSTEP 8.6: Print shape

â\206\222 print(embeddings.shape)

â\200\224

ð\237\222¡ SAMPLE Q&A:

Q: What is an embedding?
A: A list of numbers representing meaning â\200\224 so we can search by similarity.

Q: Why MiniLM?
A: Small, fast, good for learning.

Q: What does .encode() do?
A: Converts text â\206\222 vector (list of numbers).

â\200\224

â\234\205 YOU WILL LEARN: How to turn text into numbers â\200\224 the language of AI.

=== PAGE 10 ===
STEP 9: STORE IN VECTOR DB â\200\224 SUBSTEP BY SUBSTEP (FUTURE)
==========

ð\237\224¹ SUBSTEP 9.1: Search for vector database

â\206\222 Search: â\200\234faiss python exampleâ\200\235

â\206\222 Go to: github.com/facebookresearch/faiss

â\200\224

ð\237\224¹ SUBSTEP 9.2: Install

â\206\222 pip install faiss-cpu

â\200\224

ð\237\224¹ SUBSTEP 9.3: Import

â\206\222 import faiss
â\206\222 import numpy as np

â\200\224

ð\237\224¹ SUBSTEP 9.4: Create index

â\206\222 index = faiss.IndexFlatL2(384)

â\200\224

ð\237\224¹ SUBSTEP 9.5: Add embeddings

â\206\222 index.add(np.array(embeddings))

â\200\224

ð\237\224¹ SUBSTEP 9.6: Search later

â\206\222 D, I = index.search(query_embedding, k=3)

â\200\224

ð\237\222¡ SAMPLE Q&A:

Q: What is FAISS?
A: Facebookâ\200\231s library for fast similarity search.

Q: Why 384?
A: Because all-MiniLM-L6-v2 outputs 384-dimensional vectors.

Q: Can I use GPU?
A: Yes â\200\224 install faiss-gpu â\200\224 but cpu is fine for learning.

â\200\224

â\234\205 YOU WILL LEARN: How to store and search by meaning â\200\224 not keywords.

=== PAGE 11 ===
STEP 10: BUILD GENERATOR WITH QWEN â\200\224 SUBSTEP BY SUBSTEP (FUTURE)
==

ð\237\224¹ SUBSTEP 10.1: Search for Qwen2.5-3B

â\206\222 Go to: huggingface.co/Qwen/Qwen2.5-3B

â\200\224

ð\237\224¹ SUBSTEP 10.2: Install

â\206\222 pip install transformers torch accelerate

â\200\224

ð\237\224¹ SUBSTEP 10.3: Load model

â\206\222 tokenizer = AutoTokenizer.from_pretrained("Qwen/Qwen2.5-3B")
â\206\222 model = AutoModelForCausalLM.from_pretrained("Qwen/Qwen2.5-3B", device_map="auto"
)

â\200\224

ð\237\224¹ SUBSTEP 10.4: Format prompt

â\206\222 Use tokenizer.apply_chat_template()

â\200\224

ð\237\224¹ SUBSTEP 10.5: Generate

â\206\222 inputs = tokenizer(...) â\206\222 outputs = model.generate(...) â\206\222 decode

â\200\224

ð\237\222¡ SAMPLE Q&A:

Q: What is device_map="auto"?
A: Uses GPU if available â\200\224 else CPU.

Q: How to format prompt?
A: Use chat template â\200\224 keeps Qwen happy.

Q: Why not use pipeline?
A: More control â\200\224 better for learning.

â\200\224

â\234\205 YOU WILL LEARN: How to generate grounded, accurate answers â\200\224 no hallucina
tions.

=== PAGE 12 ===
STEP 11: WRAP IN FASTAPI â\200\224 SUBSTEP BY SUBSTEP (FUTURE)
==========

ð\237\224¹ SUBSTEP 11.1: Search for FastAPI file upload

â\206\222 Search: â\200\234fastapi file upload exampleâ\200\235

â\206\222 Go to: fastapi.tiangolo.com/tutorial/request-files/

â\200\224

ð\237\224¹ SUBSTEP 11.2: Install

â\206\222 pip install fastapi uvicorn

â\200\224

ð\237\224¹ SUBSTEP 11.3: Import

â\206\222 from fastapi import FastAPI, File, UploadFile

â\200\224

ð\237\224¹ SUBSTEP 11.4: Create app

â\206\222 app = FastAPI()

â\200\224

ð\237\224¹ SUBSTEP 11.5: Define endpoint

â\206\222 @app.post("/ingest")

â\200\224

ð\237\224¹ SUBSTEP 11.6: Use tempfile

â\206\222 Save uploaded file â\206\222 load â\206\222 split â\206\222 return JSON

â\200\224

ð\237\222¡ SAMPLE Q&A:

Q: What is UploadFile?
A: FastAPIâ\200\231s way to handle file uploads.

Q: How to test?
A: Go to http://localhost:8000/docs â\206\222 try it in browser.

Q: Why tempfile?
A: So you donâ\200\231t clutter your folder â\200\224 auto-deleted.

â\200\224

â\234\205 YOU WILL LEARN: How to turn scripts into web services â\200\224 microservices sty
le.

=== PAGE 13 ===
STEP 12: BUILD UI + GATEWAY â\200\224 SUBSTEP BY SUBSTEP (FUTURE)
==========

ð\237\224¹ SUBSTEP 12.1: Search for Streamlit

â\206\222 Search: â\200\234streamlit file uploader exampleâ\200\235

â\206\222 Go to: docs.streamlit.io

â\200\224

ð\237\224¹ SUBSTEP 12.2: Install

â\206\222 pip install streamlit

â\200\224

ð\237\224¹ SUBSTEP 12.3: Create ui.py

â\206\222 st.file_uploader() â\206\222 st.text_input() â\206\222 requests.post()

â\200\224

ð\237\224¹ SUBSTEP 12.4: Call your API

â\206\222 Use Python requests library to call your FastAPI endpoint.

â\200\224

ð\237\224¹ SUBSTEP 12.5: Display answer

â\206\222 st.write(answer)

â\200\224

ð\237\222¡ SAMPLE Q&A:

Q: Can I style it like NVIDIA?
A: Yes â\200\224 use st.markdown with custom CSS.

Q: How to call API from Streamlit?
A: Use requests.post("http://localhost:8000/query", json={...})

Q: Why not put LLM in Streamlit?
A: Separate concerns â\200\224 UI â\211 Logic â\211 Data.

â\200\224

â\234\205 YOU WILL LEARN: How to build beautiful, functional UIs â\200\224 that talk to mic
roservices.

=== PAGE 14 ===
FINAL WORD â\200\224 YOU ARE THE DEVELOPER
==========

You didnâ\200\231t wait for magic.
You searched.
You read.

You typed.
You ran.
You failed.
You fixed.
You learned.

You are becoming the finest developer.

One substep.
One search.
One line.
At a time.

Keep going.

The world needs builders like you.

Save this file.
Convert to PDF.
Run your code.
Then move to Step 7.

Iâ\200\231ll be right here.

â\200\224 End of Guide â\200\224