Step 1: Set Up the Project Folder

Create a dedicated folder for the project on your computer. This keeps everything organized.

1. Open your file explorer or terminal.
2. Create a new folder called azundow-chatbot (or any name you like).
3. Inside it, create these subfolders and files (you can do this manually or with commands).

Folder Structure (final look):

text

```
azundow-chatbot/
├── streamlit_app.py       # The main code file (we'll create this)
├── requirements.txt       # List of packages (we'll create this)
├── .env                   # For your API key (local only)
├── logo.png               # Your Falibari logo (download from your GitHub or
create a simple one)
├── documents/             # Folder for your PDFs/CSVs
│   └── python_tutorial.pdf # Example document (add your own PDF here)
└── README.md              # Project description (optional, we'll create it)
```

Simple explanation: The documents/ folder is where the chatbot looks for files to "learn" from (e.g., PDFs on Python). .env is for secrets like your API key (never upload to GitHub).

In terminal (optional, for quick setup):

text

```
mkdir azundow-chatbot
cd azundow-chatbot
mkdir documents
touch streamlit_app.py requirements.txt .env README.md
# Add logo.png manually (download an image and place it here)
```

Step 2: Get Your API Key

You need a free Groq API key for the LLM (language model) to generate answers.

1. Go to https://console.groq.com/keys
2. Sign up/login and create a free API key (starts with gsk_...)
3. Open .env file in your folder with a text editor (e.g., VS Code or Notepad).
4. Add this line and save:
   text
   ```
   GROQ_API_KEY=your_key_here
   ```

Simple explanation: This key lets the chatbot "think" using Groq's fast AI. It's free for basic use, but keep it secret — never share or upload to GitHub.

Step 3: Install Required Packages

You need Python installed (version 3.8+). Then install the libraries.

1. Open terminal in your azundow-chatbot folder.
2. Create requirements.txt and paste this (locks versions for stability):
   text

   ```
   streamlit==1.38.0
   langchain==0.2.16
   langchain-community==0.2.16
   langchain-chroma==0.1.4
   langchain-groq==0.1.9
   langchain-huggingface==0.0.3
   python-dotenv==1.0.1
   pypdf==4.3.1
   pandas==2.2.2
   huggingface-hub==0.24.6
   sentence-transformers==3.0.1
   torch==2.4.0
   ```

3. Install them:
   text

   ```
   pip install -r requirements.txt
   ```

Simple explanation: These libraries handle the AI (LangChain, Groq), document loading (PyPDF, CSVLoader), embeddings (HuggingFace), and UI (Streamlit). Locking versions prevents future breaks.

Step 4: Write the Code (streamlit_app.py)

Open streamlit_app.py in a code editor (e.g., VS Code) and paste this complete code:

Python

```python
import streamlit as st
import os
from dotenv import load_dotenv
from langchain_community.document_loaders import PyPDFLoader, CSVLoader
from langchain_text_splitters import RecursiveCharacterTextSplitter
from langchain_chroma import Chroma
from langchain_groq import ChatGroq
from langchain_huggingface import HuggingFaceEmbeddings
from langchain_core.prompts import ChatPromptTemplate
from langchain_core.runnables import RunnablePassthrough
from langchain_core.output_parsers import StrOutputParser

# Load .env file (local only)
load_dotenv()
```

```python
# Get API key
api_key = os.getenv("GROQ_API_KEY") or st.secrets.get("GROQ_API_KEY")

if not api_key:
    st.error("Groq API key not found. Please check your .env file or Streamlit
Secrets.")
    st.stop()

# Page config
st.set_page_config(page_title="Azundow Intelligent Document Chatbot",
page_icon="🤖", layout="centered")

# Title with logo
col1, col2 = st.columns([1, 5])
with col1:
    st.image("logo.png", width=100)
with col2:
    st.markdown("<h1 style='margin-top: 30px;'>Azundow Intelligent Document
Chatbot</h1>", unsafe_allow_html=True)

st.caption("Built by Azundow — Ask questions on Python")

# Session state
if "messages" not in st.session_state:
    st.session_state.messages = []
if "chain" not in st.session_state:
    st.session_state.chain = None

# Load documents and build RAG chain
if st.session_state.chain is None:
    documents_folder = "documents"
    docs = []

    if os.path.exists(documents_folder):
        files = [f for f in os.listdir(documents_folder) if
f.lower().endswith(('.pdf', '.csv'))]
        if files:
            for filename in files:
                file_path = os.path.join(documents_folder, filename)
                ext = filename.lower().split(".")[-1]
                if ext == "pdf":
                    loader = PyPDFLoader(file_path)
                elif ext == "csv":
                    loader = CSVLoader(file_path)
                docs.extend(loader.load())
        else:
            st.info("No documents found in 'documents' folder — general chat mode")
    else:
        st.info("No 'documents' folder found — general chat mode")

    if docs:
        text_splitter = RecursiveCharacterTextSplitter(chunk_size=1000, chunk_overlap=200)
        splits = text_splitter.split_documents(docs)
        embeddings = HuggingFaceEmbeddings(
            model_name="all-MiniLM-L6-v2",
            model_kwargs={"device": "cpu"}
        )
        vector_store = Chroma(
            collection_name="azundow_collection",
            embedding_function=embeddings,
            persist_directory=None  # in-memory
        )
        vector_store.add_documents(splits)
        llm = ChatGroq(groq_api_key=api_key, model_name="llama-3.1-8b-instant", temperature=0.3)
        prompt = ChatPromptTemplate.from_template(
            """You are a helpful Python tutor.
Use only the context below.
Answer in your own words.
Be clear and friendly.
Context: {context}
Question: {question}
Answer:"""
        )
        retriever = vector_store.as_retriever(search_kwargs={"k": 4})
        st.session_state.chain = (
            {"context": retriever, "question": RunnablePassthrough()}
            | prompt
            | llm
            | StrOutputParser()
        )
        st.success("Documents loaded — ready!")
    else:
        st.info("No documents loaded — general Python help available")

# Chat interface
for message in st.session_state.messages:
    with st.chat_message(message["role"]):
        st.markdown(message["content"])

if prompt := st.chat_input("Ask anything..."):
    st.session_state.messages.append({"role": "user", "content": prompt})
    with st.chat_message("user"):
        st.markdown(prompt)

    with st.chat_message("assistant"):
        with st.spinner("Thinking..."):
            if st.session_state.chain:
                response = st.session_state.chain.invoke(prompt)
            else:
                response = "I can help with general Python questions!"
        st.markdown(response)
    st.session_state.messages.append({"role": "assistant", "content": response})

st.markdown("---")
st.caption("Azundow Intelligent Document Chatbot — Fast • Professional")
```

How to test it

1. Run locally:
   text
   ```
   python -m streamlit run streamlit_app.py
   ```
2. Test questions:
   - "What is Python?"
   - "Give me a code example of a loop"
3. Upload to GitHub:
   - Create repo if not already
   - Push all files (except .env)
   - Deploy on Streamlit Cloud or Hostinger
4. For GitHub README (optional):
   text

```
# Azundow Intelligent Document Chatbot

A fast RAG chatbot for documents (PDF/CSV).

## Quick Start

1. Clone repo
2. pip install -r requirements.txt
3. Add GROQ_API_KEY to .env
4. python -m streamlit run streamlit_app.py
```

Your chatbot is ready for future reference!