```
import streamlit as st
import pandas as pd
from PIL import Image
import os
from utils.file_validator import FileValidator
from utils.file_processor import FileProcessor

# Configure page
st.set_page_config(
    page_title="File Upload Center",
    page_icon="📁",
    layout="wide",
    initial_sidebar_state="expanded"
)

def main():
    st.markdown(
        """
        <div style="background: linear-gradient(90deg, #667eea 0%, #764ba2 100%);
                    padding: 2rem;
                    border-radius: 10px;
                    margin-bottom: 2rem;
                    text-align: center;
                    color: white;
                    box-shadow: 0 4px 6px rgba(0, 0, 0, 0.1);">
            <h1 style="margin: 0; font-size: 3rem; font-weight: bold;">
                File Upload Center
            </h1>
            <p style="margin: 0.5rem 0 0 0; font-size: 1.2rem; opacity: 0.9;">
                Upload and manage your files
```

```
        </p>
    </div>
    """,
    unsafe_allow_html=True
)

# Upload Section
uploaded_files = st.file_uploader(
    "Choose files to upload",
    accept_multiple_files=False,  # for chatbot, handle one file at a time
    help="Upload a file to interact with via chatbot",
    type=None
)

if uploaded_files:
    process_uploaded_file(uploaded_files, 0)



    # Initialize chatbot history
    if "messages" not in st.session_state:
        st.session_state.messages = []

    st.markdown("### 💬 Chat with your file")

    # Display previous chat
    for msg in st.session_state.messages:
        with st.chat_message(msg["role"]):
            st.markdown(msg["content"])

    # User input
```

```
        if user_query := st.chat_input("Ask me something about this file..."):
            st.session_state.messages.append({"role": "user", "content": user_query})

            # Generate response
            response = answer_about_file(user_query, uploaded_files)
            st.session_state.messages.append({"role": "assistant", "content": response})

            # Display response
            with st.chat_message("assistant"):
                st.markdown(response)

def process_uploaded_file(uploaded_file, index):
    """Process an individual uploaded file"""
    validator = FileValidator()
    processor = FileProcessor()

    validation_result = validator.validate_file(uploaded_file)

    if validation_result['is_valid']:
        file_info = processor.process_file(uploaded_file, validation_result['category'])
        st.success(f"✅ File '{uploaded_file.name}' uploaded successfully!")

        # Save file_info in session for chatbot use
        st.session_state["file_info"] = file_info

    else:
        st.error(f"❌ Upload failed: {validation_result['error']}")

def answer_about_file(query, uploaded_file):
    """
```

```
    Simple Q&A over uploaded file.
    For now: keyword-based responses using file metadata.
    """
    file_info = st.session_state.get("file_info", {})

    if not file_info:
        return "I don’t have any details about the file yet."

    # Very basic logic (you can replace with LLM later)
    if "name" in query.lower():
        return f"The file name is **{file_info['file_name']}**."
    elif "size" in query.lower():
        return f"The file size is **{file_info['file_size_bytes'] / (1024*1024):.2f} MB**."
    elif "type" in query.lower() or "category" in query.lower():
        return f"The file category is **{file_info['category']}**."
    elif "info" in query.lower() or "details" in query.lower():
        return f"Here’s what I know: {file_info}"
    else:
        return "I can currently answer about file name, size, type, or details. Try asking about
those!"

if __name__ == "__main__":
    main()
```