ComputerHubMarket Task

1. Import Necessary Libraries

- **Streamlit** for creating the web application interface.
- **PyPDF2** and **PyPDFLoader** for loading and processing PDF files.
- **LangChain** components for text processing and question answering.
- **Google Generative AI** for embedding and conversational AI functionalities.
- **FAISS** for building and managing a vector store for efficient similarity search.
- **dotenv** for loading environment variables.
- **NamedTemporaryFile** for handling temporary files.

2. Load Environment Variables

- Use `dotenv` to load environment variables from a `.env` file.
- Retrieve the `GOOGLE_API_KEY` and configure Google Generative AI with it. If the key is not set, display an error message.

3. PDF Loading and Processing

- **Load PDFs:** Define a function `load_pdf` that reads and loads content from uploaded PDF files using `PyPDFLoader`.
- **Split Text into Chunks:** Define a function `get_text_chunks` that concatenates text from PDF pages and splits it into manageable chunks using `RecursiveCharacterTextSplitter`.

4. Create Vector Store

- **Generate Embeddings:** Use `GoogleGenerativeAIEmbeddings` to create embeddings from the text chunks.
- **Build FAISS Index:** Use FAISS to create a vector store from the text chunks and save it locally for later retrieval.

5. Create Conversational Chain

- Define a function `get_conversational_chain` that sets up a question-answering chain using `ChatGoogleGenerativeAI` and a custom prompt template. The chain ensures that answers are provided based on the context or states that the answer is not available.

6. Handle User Input

- Define a function `user_input` that handles the user's question.
- Load the FAISS index and search for similar documents based on the user's question.
- Use the conversational chain to generate a detailed answer from the found documents and display it.

7. Build Streamlit Interface

- Set up the Streamlit page configuration and header.
- Add a text input field for users to ask questions.
- Handle PDF file uploads through the sidebar.
- On submitting the PDFs, process them to extract text, split it into chunks, and build the vector store.
- If no PDF is uploaded, display an error message.

8. Run the Application

- Define the `main` function to bring all components together and run the Streamlit app.
- If the script is executed directly, call the `main` function to start the app.

Summary of the Flow:

1. **User uploads PDF files.**
2. **System processes PDFs:**
    - Loads text from PDFs.
    - Splits text into chunks.
    - Creates a vector store using FAISS.
3. **User inputs a question:**
    - System retrieves relevant text chunks from the vector store.
    - Uses a conversational AI chain to generate and display a detailed answer.