* + **1. Start:**
  + **Shape**: Parallelogram
  + **Text**: "User Query"
  + **Description**: The user inputs a query like "I want to learn machine learning."
  + **Flow**: Single arrow pointing to the next step.
  + **2. Encode User Query (Linear Step):**
  + **Shape**: Rectangle
  + **Text**: "Encode User Query (with Sentence-BERT)"
  + **Description**: The user's query is encoded into an embedding vector for further retrieval.
  + **Flow**: Single arrow pointing to the next step.
  + **3. Data Preparation (Parallel Step, Broken into Sub-Steps):**
  + **Shape**: Rectangle
  + **Text**: "Data Preparation"
  + **Description**: This block will contain multiple parallel tasks related to preparing the dataset.
  + **Sub-step 3.1**: **Load Dataset**  
    **Shape**: Rectangle  
    **Text**: "Load and Clean Data"  
    **Description**: Load the CSV or dataset containing course information and handle NaN values.
  + **Sub-step 3.2**: **Combine Text Fields**  
    **Shape**: Rectangle  
    **Text**: "Combine Title, Short Intro, Skills"  
    **Description**: Combine text fields into a combined\_text column.
  + **Sub-step 3.3**: **Preprocessing**  
    **Shape**: Rectangle  
    **Text**: "Text Preprocessing"  
    **Description**: Optional step to preprocess text (cleaning, tokenization, etc.).
  + **Flow**: All **sub-steps are parallel** (draw each rectangle side by side with arrows pointing to the next major step).
  + **4. Generate Course Embeddings (Linear Step):**
  + **Shape**: Rectangle
  + **Text**: "Generate Course Embeddings"
  + **Description**: Use a pre-trained model (Sentence-BERT) to encode the combined course data into vectors.
  + **Flow**: Single arrow pointing to the next step.
  + **5. Create FAISS Index (Linear Step):**
  + **Shape**: Rectangle
  + **Text**: "Create FAISS Index"
  + **Description**: Create a FAISS index to store and retrieve the embedded vectors for course descriptions.
  + **Flow**: Single arrow pointing to the next step.
  + **6. Retrieve Relevant Courses (Similarity Search) (Linear Step):**
  + **Shape**: Rectangle
  + **Text**: "Retrieve Relevant Courses"
  + **Description**: Use the FAISS index to find the top N most similar courses to the encoded user query.
  + **Flow**: Single arrow pointing to the next step.
  + **7. Augment Query with Retrieved Courses (Linear Step):**
  + **Shape**: Rectangle
  + **Text**: "Augment Query with Retrieved Courses"
  + **Description**: Concatenate the user query with the retrieved course descriptions to create a context-rich input for the generation step.
  + **Flow**: Single arrow pointing to the next step.
  + **8. Response Generation (LLM) (Linear Step):**
  + **Shape**: Rectangle
  + **Text**: "Generate Response Using LLM"
  + **Description**: Use an LLM (like GPT-3) to generate a personalized response based on the augmented query and course information.
  + **Flow**: Single arrow pointing to the next step.
  + **9. Chatbot Framework (Conversation Management) (Parallel Step):**
  + **Shape**: Rectangle
  + **Text**: "Chatbot Framework"
  + **Description**: This is where you manage the conversation flow (input, output, retry queries, etc.).
  + **Sub-step 9.1**: **Handle Conversation**  
    **Shape**: Rectangle  
    **Text**: "Handle Conversation Flow"  
    **Description**: Make sure the conversation stays on track, handling user input, output, and retries.
  + **Sub-step 9.2**: **Send Response to User**  
    **Shape**: Rectangle  
    **Text**: "Send Response to User"  
    **Description**: Send the LLM-generated personalized recommendation back to the user.
  + **Flow**: Both **sub-steps are parallel**, but eventually lead to the same end.
  + **10. Deployment & Scalability (Parallel Step):**
  + **Shape**: Rectangle
  + **Text**: "Deployment & Scalability"
  + **Description**: Ensure the chatbot is scalable and ready for a cloud environment.
  + **Sub-step 10.1**: **Host on Cloud**  
    **Shape**: Rectangle  
    **Text**: "Host Chatbot on Cloud Platform"  
    **Description**: Deploy the chatbot on AWS, GCP, or Azure to scale and handle more users.
  + **Sub-step 10.2**: **Implement Caching**  
    **Shape**: Rectangle  
    **Text**: "Use Caching for Fast Retrieval"  
    **Description**: Cache retrieved courses or embeddings to improve response time.
  + **Flow**: These are **parallel tasks** but will ensure smooth user experience and scalability.
  + **11. End:**
  + **Shape**: Parallelogram
  + **Text**: "Response Delivered to User"
  + **Description**: The system sends back the course recommendation response to the user based on their query.
  + **How to Structure the Diagram:**
  + **Start** with the **User Query** at the top (parallelogram).
  + **Linear Process**:
  + **Encode User Query** → **Data Preparation (with parallel sub-steps)** → **Generate Embeddings** → **Create FAISS Index** → **Retrieve Relevant Courses** → **Augment Query** → **Generate Response Using LLM**.
  + **Parallel Blocks**:
  + **Chatbot Framework** and **Deployment & Scalability** steps should be drawn side-by-side to show that they are parallel processes that run simultaneously.
  + **End** the process with **Response Delivered to User** (parallelogram).
  + **Key Notes for Drawing:**
  + **Use arrows** to indicate the flow of data from one step to another.
  + **Parallel processes** (like Data Preparation sub-steps and Chatbot Framework/Deployment tasks) should be **side by side**.
  + **Each box/shape** (rectangle for steps, parallelogram for user input/output) should have a short description.