**1. Create & Design Your Canvas App**

* In Power Apps, select **Canvas app from blank**, name it “EduHackAI Bot.”
* Add controls:
  + **Text input** (e.g., txtQuestion) for learner queries.
  + **Button** (btnAsk) labeled “Ask.”
  + **Gallery** (galAnswers) or **Labels** to display AI responses.
* Initialize state on the app’s **OnStart** property:

powerfx

Set(chatLog, []);

**2. Build a Custom Connector**

* Navigate to **Dataverse > Custom connectors** in the Power Platform admin center.
* Click **+ New custom connector** and choose **Import an OpenAPI file** or **Create from blank**.
* Define:
  + **Host URL**: your Azure Function or API Gateway endpoint
  + **Security**: API key or Azure AD
  + **Request schema**: { "prompt": "string" }
  + **Response schema**: { "answer": "string" }
* Test connection and save.

**3. Create a Power Automate Flow**

* In Power Automate, choose **+ New → Instant cloud flow**, trigger: **Power Apps**.
* Add the **Custom Connector** action or **HTTP** action pointing to your AI endpoint:
  + Method: POST
  + Body:

json

{

"prompt": "@{triggerBody()['text']}"

}

* Add **Respond to Power Apps** action, passing back the answer field.

**4. Embed & Invoke the Flow in Canvas App**

* In your Canvas app, under **Data**, add the flow you just created.
* Update the **OnSelect** of btnAsk:

powerfx

// Capture user question

Collect(chatLog, { from: "user", text: txtQuestion.Text });

// Call backend flow

Set(aiResponse, EduHackFlow.Run(txtQuestion.Text));

// Append AI answer

Collect(chatLog, { from: "bot", text: aiResponse.answer });

* Bind galAnswers.Items to chatLog and format each item with conditional styling based on from.