**Day 4 – GitHub Steps & Power Apps Kick-off**

Below are the **exact GitHub commands and file edits** for Day 4, followed by the **Power Apps/Automate starting steps** (flow stub) to get your bot talking.

**GitHub Steps**

1. Open your function handler and make it echo input:
   * File: azure-functions/ChatHandler.cs
   * Replace with:

csharp

[FunctionName("ChatHandler")]

public static async Task<IActionResult> Run(

[HttpTrigger(AuthorizationLevel.Function, "post")] HttpRequest req)

{

var reqBody = await new StreamReader(req.Body).ReadToEndAsync();

return new OkObjectResult(new { answer = $"Echo: {reqBody}" });

}

1. Add a basic xUnit test harness:
   * Path: azure-functions/chatHandler.Tests/ChatHandlerTests.cs
   * Sample test:

csharp

public class ChatHandlerTests

{

[Fact]

public async Task ReturnsEcho()

{

var req = new DefaultHttpRequest(new DefaultHttpContext())

{

Body = new MemoryStream(Encoding.UTF8.GetBytes("Hello"))

};

var result = await ChatHandler.Run(req, NullLoggerFactory.Instance.CreateLogger("Test"));

var ok = Assert.IsType<OkObjectResult>(result);

dynamic body = ok.Value;

Assert.Equal("Echo: Hello", (string)body.answer);

}

}

1. Stage, commit & push to your feature branch:

bash

git checkout feature/ai-flow

git add azure-functions/ChatHandler.cs \

azure-functions/chatHandler.Tests/ChatHandlerTests.cs

git commit -m "Day 4: Implement ChatHandler stub + unit test"

git push origin feature/ai-flow

**Power Apps / Power Automate Start**

1. Create an Instant Cloud Flow
   * Go to https://flow.microsoft.com → **+ New → Instant cloud flow**
   * Trigger: **Power Apps**
   * Name it Day4\_WahyBot\_FlowStub
2. Add an **HTTP** action
   * Method: POST
   * URL:
   * https://<your-func-app>.azurewebsites.net/api/ChatHandler?code=<function\_key>
   * Headers:
   * Content-Type: application/json
   * Body:

json

{

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

}

1. Respond back to Power Apps
   * Add **Respond to Power Apps** action
   * Return the HTTP action’s answer field as an output parameter named answer
2. Save & share your flow
   * Click **Save**
   * Toggle **Run-only permissions** so your Canvas app can call it

Now your function stub is live, tested via GitHub CI, and your flow is ready to hook into Power Apps. Next, you’ll add this flow to your Canvas app and wire up a **Send** button to call it.

and day 3 linkeid post and 4 please seperated

**Day 3 LinkedIn Post**

**Hook** Day 3 of #MSFTHackathon: Laying the AI Foundations! 🤖

**Body**

* In GitHub, created the feature/ai-stub branch and scaffolded my /azure-functions folder with a Hello World HTTP trigger.
* Provisioned an Azure OpenAI resource in the portal—grabbed my endpoint and key to secure calls.
* Pushed the function stub to GitHub and verified the commit history for full traceability.
* Fired off a Postman request against my local function to confirm request/response plumbing.

**Visuals**

* Screenshot of the azure-functions folder tree in my repo
* Postman response showing a successful 200 OK with stub output

**Call to Action** What tips do you have for refining initial AI prompts and securing API keys?

**Links & Hashtags** Branch: https://github.com/Ghadiraldesouky/wahybot-studio/tree/feature/ai-stub #MSFTHackathon #AzureOpenAI #GitHub #DevOps #PowerPlatform

**Day 4 LinkedIn Post**

**Hook** Day 4 of #MSFTHackathon: Building WahyBot’s Chat Core & Automation! 🚀

**Body**

* Switched to feature/ai-flow: implemented ChatHandler in C# to echo user prompts and added xUnit tests for CI readiness.
* Committed and pushed both the function code and test project—our CI pipeline now has something to validate.
* In Power Automate, spun up an Instant Flow (Day4\_WahyBot\_FlowStub) with an HTTP action calling our Azure Function.
* Set up the **Respond to Power Apps** step so the flow can return answer back to the Canvas app.

**Visuals**

* Diff view of ChatHandler.cs showing the echo logic
* Screenshot of the Flow designer with HTTP action and response block

**Call to Action** Curious: how would you structure your flow outputs for richer error handling?

**Links & Hashtags** Branch: https://github.com/Ghadiraldesouky/wahybot-studio/tree/feature/ai-flow #MSFTHackathon #PowerAutomate #AzureFunctions #UnitTesting #LowCodeAI