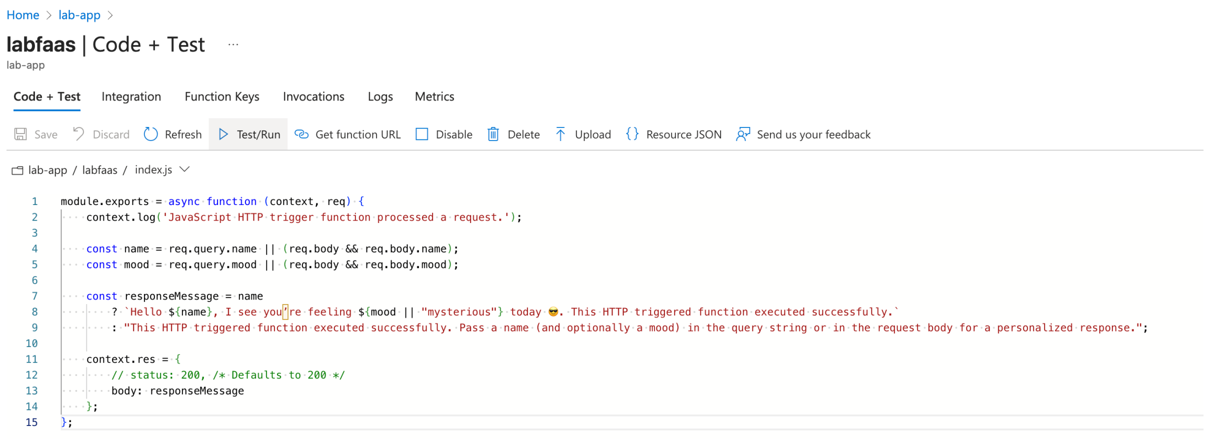
Unit 10 – Formative Activity

This post demonstrates the deployment and configuration of a serverless Azure Function using JavaScript. The function is implemented as an HTTP-triggered endpoint capable of accepting query parameters and returning a dynamic response. Specifically, the function is designed to receive name and mood parameters from an HTTP request, process them within the Node.js runtime, and respond with a personalised greeting message.

The function is deployed within an Azure Function App configured for Node.js. The code is structured as an asynchronous module export, following Azure's standard module.exports = async function (context, req) pattern. Input parameters are retrieved via req.query or req.body, allowing both GET and POST requests to be supported. Conditional logic is applied to generate a customised response based on the presence or absence of input parameters. The function returns the message via context.res.body, leveraging Azure's built-in response handling.



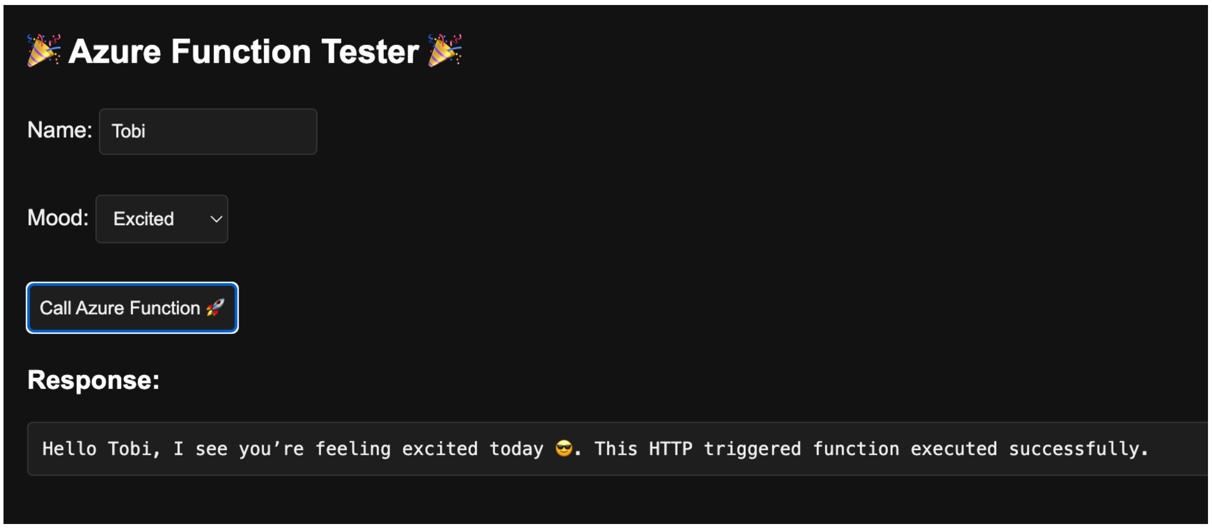
```
module.exports = async function (context, req) {
    context.log('JavaScript HTTP trigger function processed a request.');

    const name = req.query.name || (req.body && req.body.name);
    const mood = req.query.mood || (req.body && req.body.mood);

    const responseMessage = name
        ? `Hello ${name}, I see you're feeling ${mood || "mysterious"} today 😎. This HTTP triggered function executed successfully.`
        : "This HTTP triggered function executed successfully. Pass a name (and optionally a mood) in the query string or in the request body for a personalized response.";

    context.res = {
        // status: 200, /* Defaults to 200 */
        body: responseMessage
    };
};
```

To facilitate interactive testing, a front-end HTML interface was created. This interface consists of input fields for the name and mood parameters, a submit button, and a response display area. JavaScript running in the browser uses the fetch() API to call the Azure Function with the specified query parameters.



```
Hello Tobi, I see you're feeling excited today 😎. This HTTP triggered function executed successfully.
```

Critical configurations include Cross-Origin Resource Sharing (CORS) settings to permit requests from the testing origin and an authorization level set to Anonymous to allow browser access without requiring a function key. These settings ensure that the function can be invoked from any client-side environment securely and reliably.

Overall, this setup illustrates a complete technical pipeline from serverless function deployment to front-end interaction, demonstrating parameter handling, HTTP request processing, and cloud-hosted serverless architecture in Azure.

My FaaS can be testet here: https://labfaas.netlify.app/