

Project Design Documentation

Overview:

Solution 1: Actionable Message

Overview

The actionable message is a technology that allows us to embed interactive elements like buttons (thumbs-up/thumbs-down), comments, etc., into our outlook emails without needing to open a new tab to visit a website. Figure 1 below shows the example of a poll in Outlook email.

Example: Poll in Outlook email

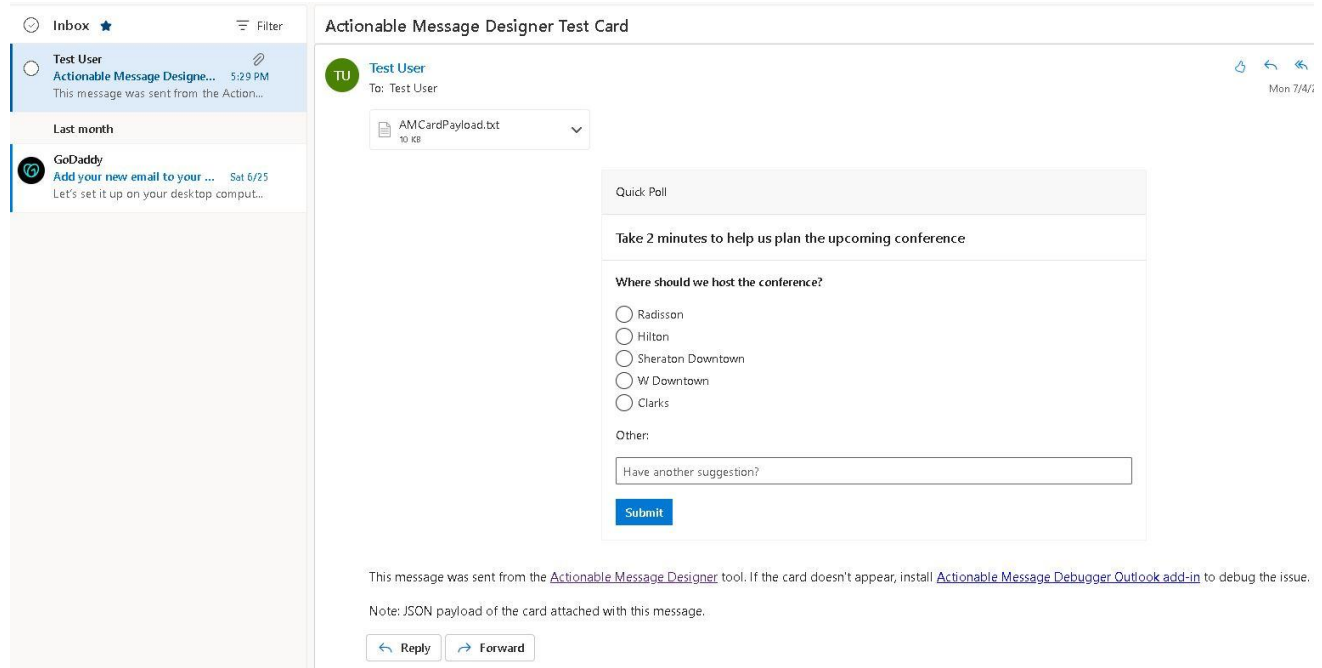


Figure 1: A poll in Outlook email

Design Steps

Step 1: Use the JSON code to build the Actionable Message card based on the given Mockup

Explain: Outlook Actionable Messages cards are designed using the Adaptive Card format. And the Adaptive Card is designed in JSON code.

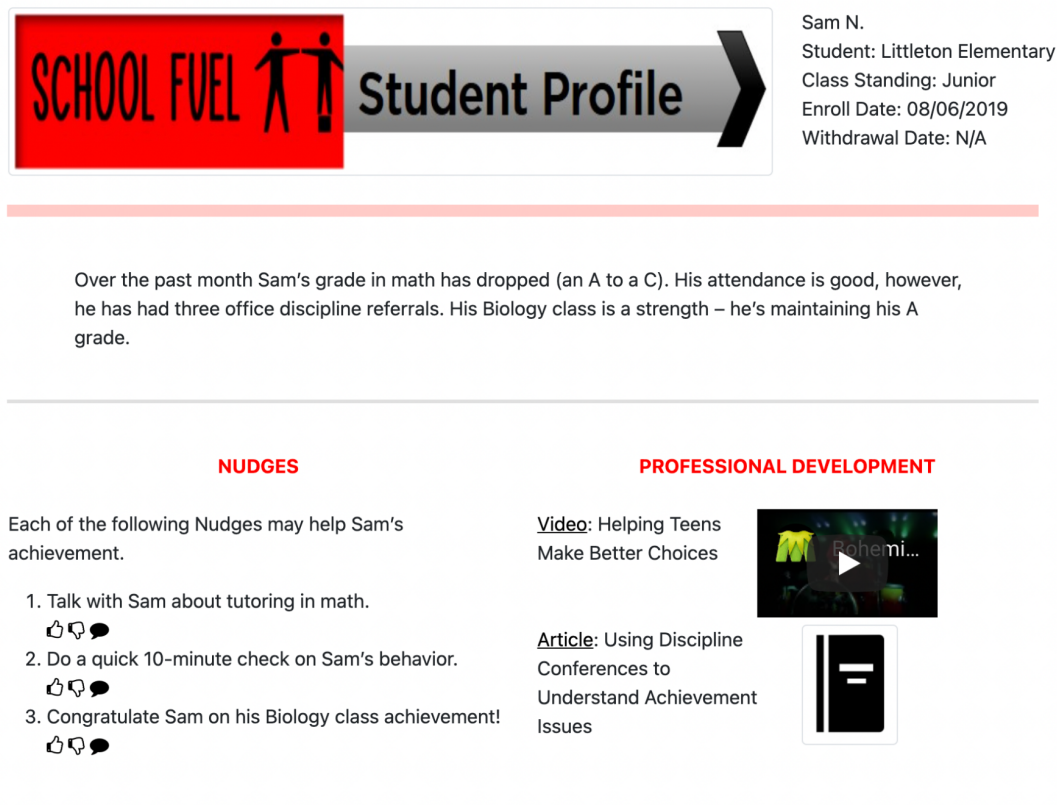


Figure 1.1: Mockup

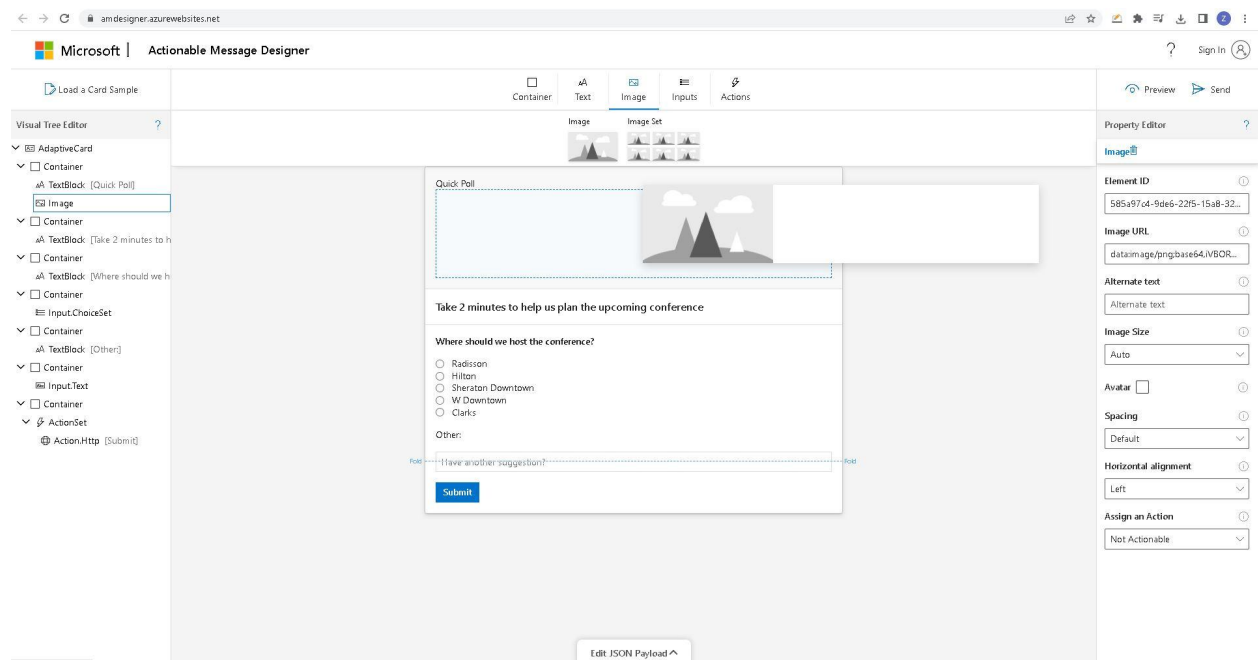
Two Approach to do Step 1:

Approach	Advantage	Disadvantage
----------	-----------	--------------

Only rewrites the Nudges part in JSON.	Save time	Not guarantee to return the same structure as the mockup
Rewrite the entire emails in JSON.	guarantee to return the same structure as the mockup	Takes more time

Tool: Adaptive Card Designer

We can use the Adaptive Cards Designer to create the cards. Adaptive Card Designer is a tool that provides a drag-and-drop experience to build and tweak adaptive cards quickly. It can also generate the JSON code.



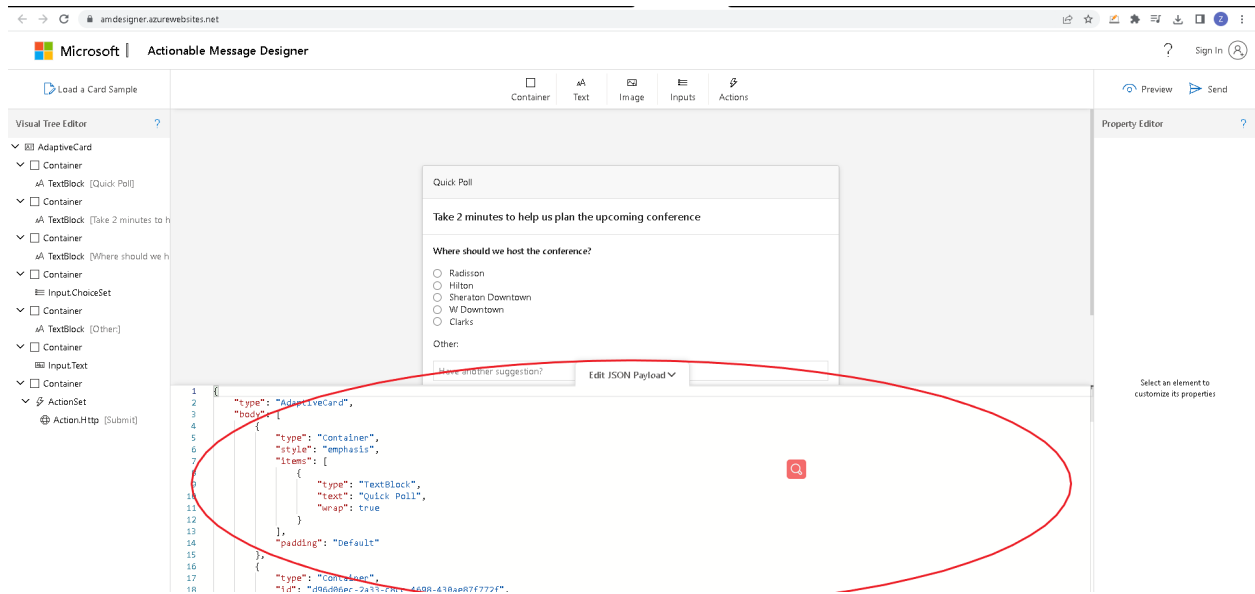
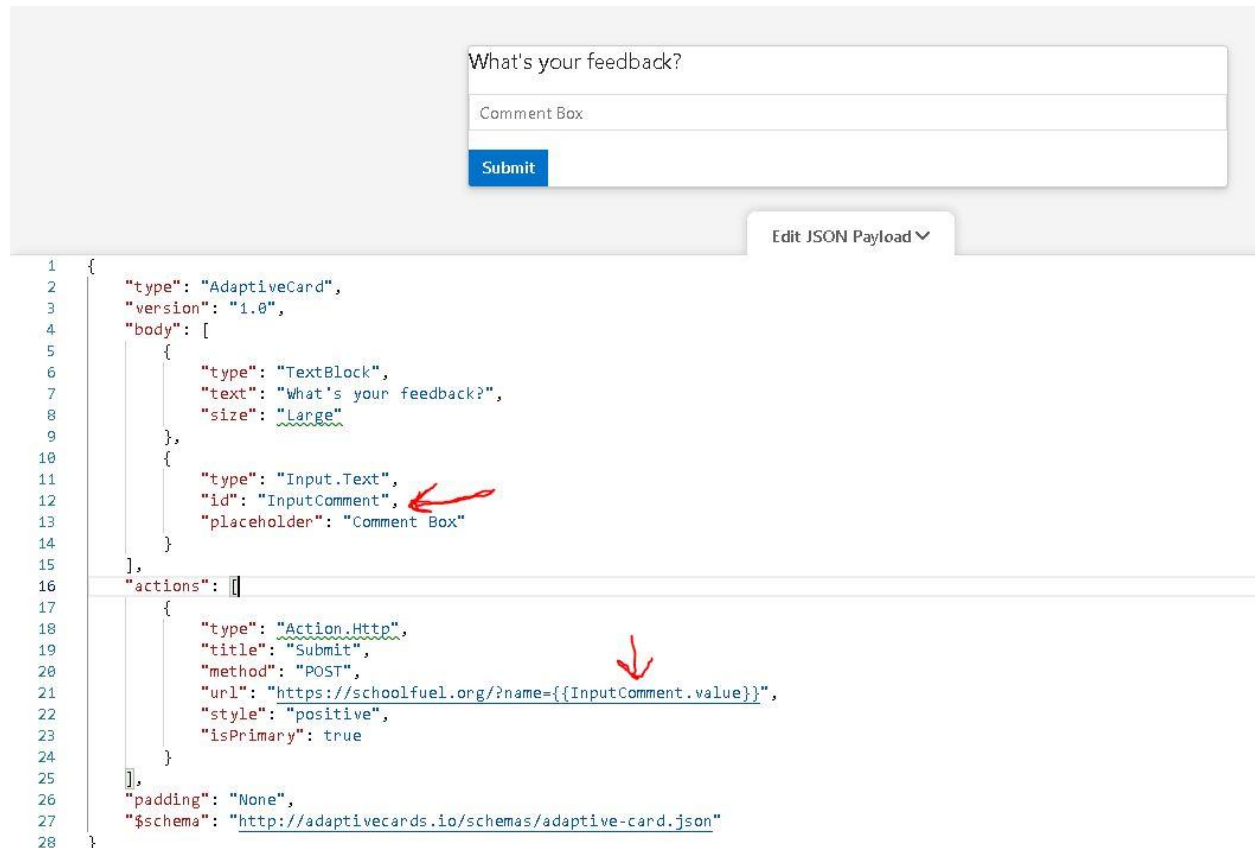


Figure 1.2: Adaptive Card Designer

Step 2: Add the target URL to the Action Button

Explain: Adaptive Cards contain inputs, and it is necessary to pass the values of these inputs to the target endpoint via an Action.Http action. This is done using input value substitution.



The image shows a visual representation of an Adaptive Card at the top and its corresponding JSON payload below. The visual card has a text input field with the placeholder text "What's your feedback?", a "Comment Box" label, and a blue "Submit" button. Below the card is a JSON payload editor with a dropdown menu set to "Edit JSON Payload". The JSON payload is as follows:

```
1 {
2   "type": "AdaptiveCard",
3   "version": "1.0",
4   "body": [
5     {
6       "type": "TextBlock",
7       "text": "What's your feedback?",
8       "size": "Large"
9     },
10    {
11      "type": "Input.Text",
12      "id": "InputComment",
13      "placeholder": "Comment Box"
14    }
15  ],
16  "actions": [
17    {
18      "type": "Action.Http",
19      "title": "Submit",
20      "method": "POST",
21      "url": "https://schoolfuel.org/?name={{InputComment.value}}",
22      "style": "positive",
23      "isPrimary": true
24    }
25  ],
26  "padding": "None",
27  "$schema": "http://adaptivecards.io/schemas/adaptive-card.json"
28 }
```

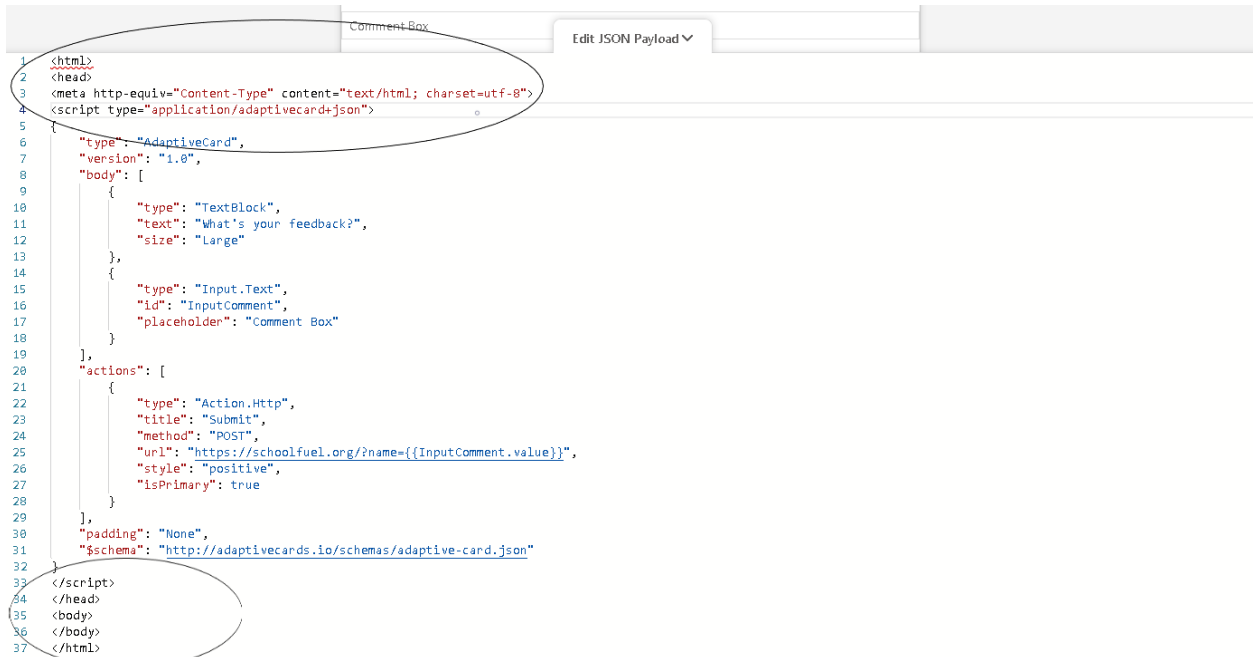
Red arrows point to the `"id": "InputComment"` property in the input definition (line 12) and the `"url": "https://schoolfuel.org/?name={{InputComment.value}}"` property in the action definition (line 21), illustrating the value substitution.

Figure 2.1 Adding Target URL

The above card defines a text input and sets its id property to "InputComment". It also defines an Action.Http action that makes a GET call to an endpoint on domain schoolfuel.org. With the inclusion of `?name={{InputComment.value}}` on the target URL, the value of the input with id "InputComment" will be dynamically substituted at the time the action is taken by the user. So if the user had entered the comment "hello" in the text input, the target URL after substitution would be `https://schoolfuel.org/?name=hello`

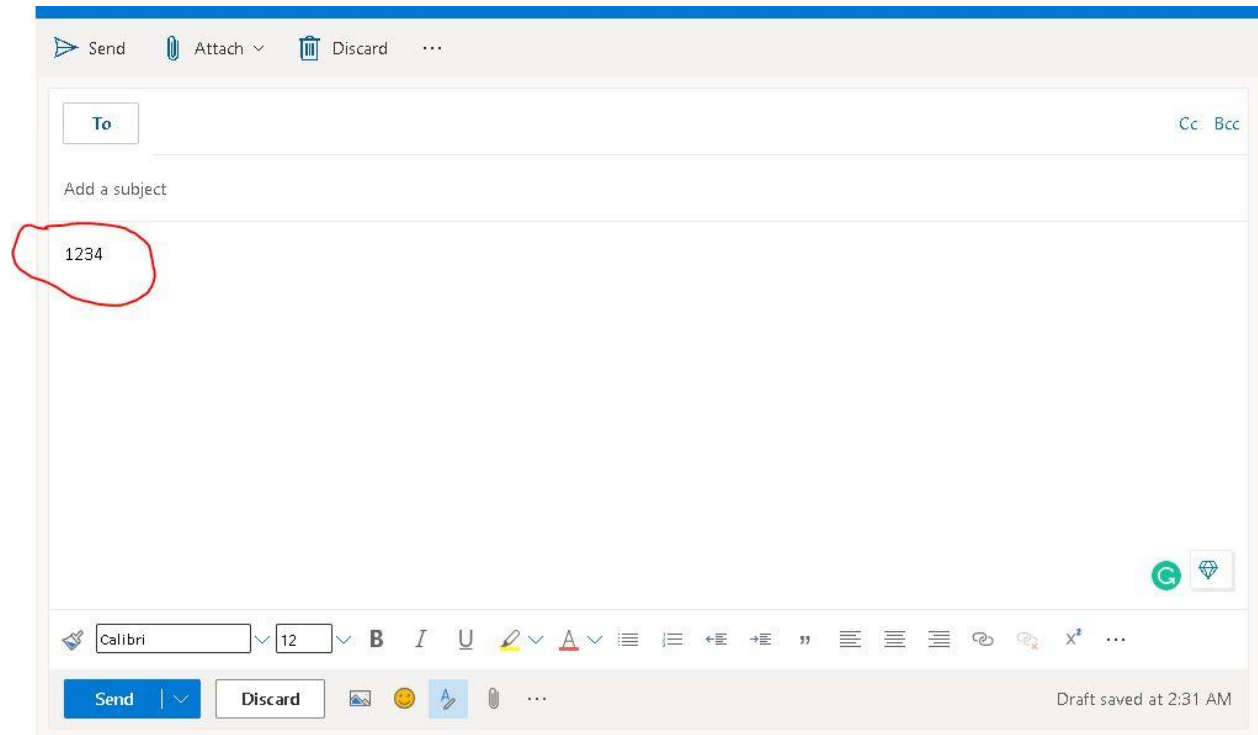
Step 3: Test/Send the built actionable message card via Outlook email

- a. To embed an actionable message card in an email message, we need to wrap the card in a <script> tag. The <script> tag is then inserted into the <head> of the email's HTML body.

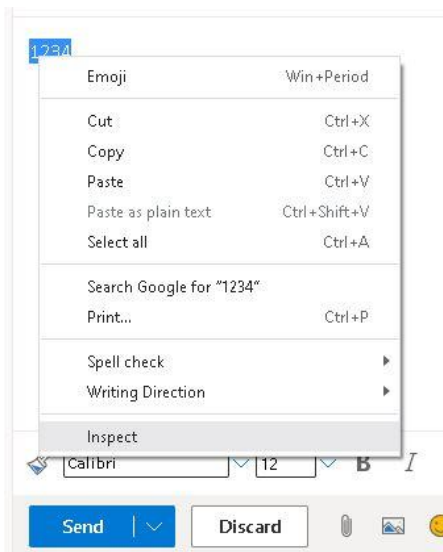


```
1 <html>
2 <head>
3 <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
4 <script type="application/adaptivecard+json">
5 {
6   "type": "AdaptiveCard",
7   "version": "1.0",
8   "body": [
9     {
10      "type": "TextBlock",
11      "text": "What's your feedback?",
12      "size": "Large"
13     },
14     {
15      "type": "Input.Text",
16      "id": "InputComment",
17      "placeholder": "Comment Box"
18     }
19   ],
20   "actions": [
21     {
22      "type": "Action.Http",
23      "title": "Submit",
24      "method": "POST",
25      "url": "https://schoolfuel.org/?name={{InputComment.value}}",
26      "style": "positive",
27      "isPrimary": true
28     }
29   ],
30   "padding": "None",
31   "$schema": "http://adaptivecards.io/schemas/adaptive-card.json"
32 }
33 </script>
34 </head>
35 <body>
36 </body>
37 </html>
```

- b. We just copy this entire bit of code. Then we go to the outlook email. Here we want to add placement "1234" to the body of the email. The placement can be anything.



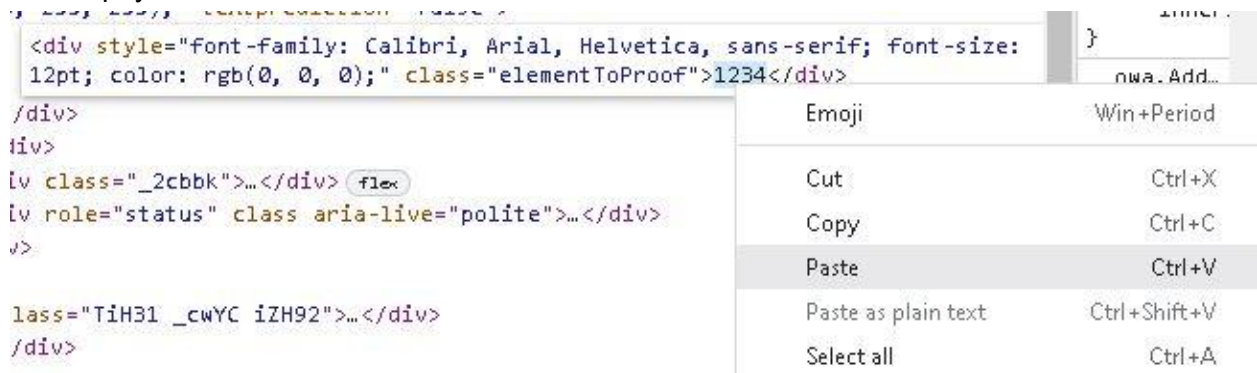
c. Highlight the placement, then right-click and inspect.



d. Navigate up to the highlight section, then right-click and choose "Edit as HTML"



- e. Navigate down to the placement “1234”, and replace it with the code from the JSON payload. And click out of that.



- f. Send the email to the same email address. For example testuser@schoolfuel.org to testuser@schoolfuel.org (FYI: you can only see the actionable message card from the inbox of outlook email)

Step 4: Check if Google Analytics can receive the data.

We have added the campaign parameters to the target URL in step 2, and we can use Google Analytics to track Custom Campaigns. Google Analytics is a tool that allows us to track everything that's happening on our website.

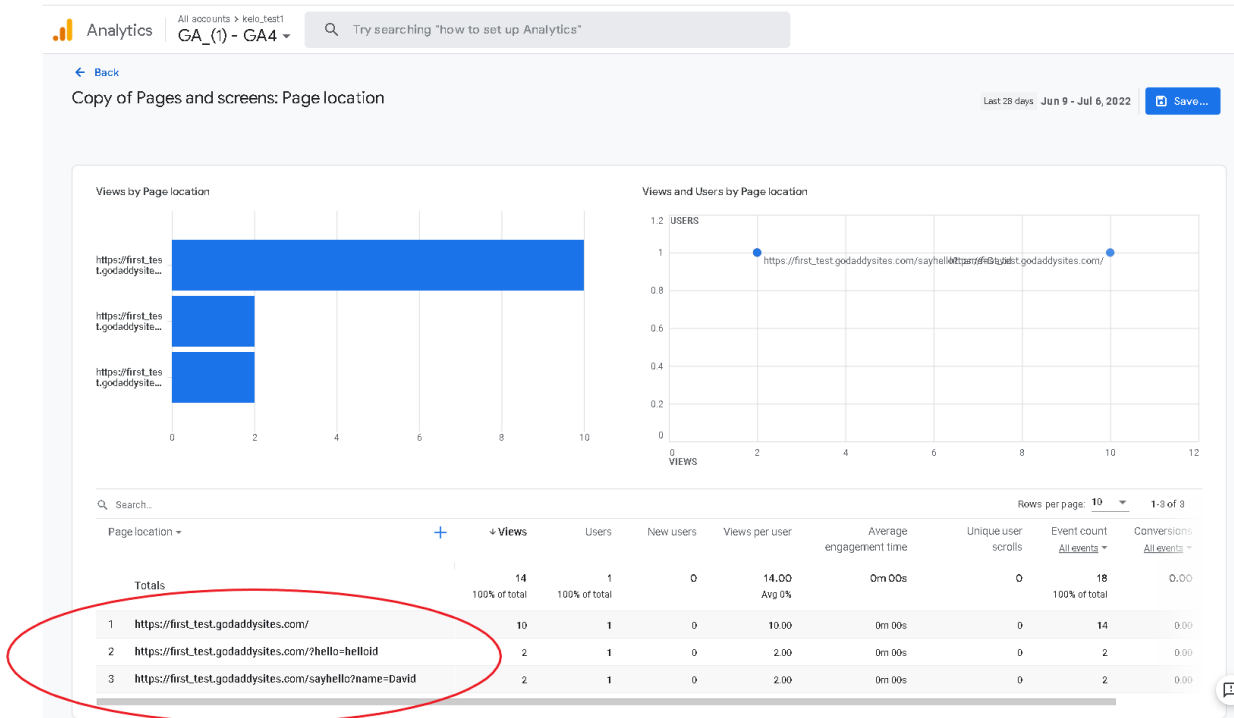


Figure 4.1: Google Analytics

Alternative way: Microsoft Power Automate

If Google Analytics does not work, we can use a tool called Power Automate. Power Automate is a tool that can generate a new target URL. We can use this target URL instead of the school fuel domain.

When the actionable message sends HTTP request to Power Automate, Power Automate can get the information from the request and return it to the response object.

The image displays two steps from a Microsoft Power Automate flow, connected by a downward arrow. The top step is a trigger named 'When a HTTP request is received'. It features a text box for 'HTTP POST URL' containing the placeholder 'URL will be generated after save' and a 'Request Body JSON Schema' text area. Below these is a link 'Use sample payload to generate schema' and a 'Show advanced options' dropdown. The bottom step is an action named 'Response'. It includes a '* Status Code' field with the value '200', a 'Headers' section with 'Enter key' and 'Enter value' input boxes, and a 'Body' field with 'Enter response content'. It also has a 'Show advanced options' dropdown.

When a HTTP request is received

HTTP POST URL: URL will be generated after save

Request Body JSON Schema

[Use sample payload to generate schema](#)

Show advanced options

Response

* Status Code: 200

Headers: Enter key, Enter value

Body: Enter response content

Show advanced options

Figure 4.2: Microsoft Power Automate (Alternative way)

Step 5: Redesign the existing text locator.py script:

The function that needs to be edited

Function name	Current property	New property
def code_html()	Build HTML code	Build HTML + JSON code
def Sent_email()	Send email using Gmail	Send email using Outlook