Gherkin Best Practices General Guidelines:

# Function of the 3 arguments – Given, When, Then

1. Given – provides the background/state of the solution. This is only setting the context. We are identifying the context of the user, and a given account
   1. No assertion is being performed in this step
   2. Ex: location,
2. When – User actions, only the user! Only in the context of the current scenario
   1. No assertion is being performed in this step
   2. We should try to limit to 1 or 2 “when” arguments per scenario
   3. Ex: clicking a tile, clicking a button, hovering the mouse
3. Then – Where all of the assertions are happening, the end goal should appear.
   1. No user action in this step, this is for the results
   2. We can have as many “Thens” as we want
   3. We should be checking a positive outcome
   4. Ex: the image should disappear,

# Rules – No Exceptions

1. The order must be
   1. Given
      1. The “Given” must appear first. If multiple given statements are needed, given can be followed up by an And
   2. When
      1. The “When” must appear after the given and before the “Then”.
   3. Then
      1. The “Then” must appear after the When
2. Multiple Given’s, When’s and Then’s are acceptable, but only sequentially. We cannot create an action after an assertion. We would need a different scenario in this case to cover that additional action
   1. Ex:Bad -> We would have to split the below example up into separate scenarios

Given…

And…

When…

And…

Then…

When

Then

1. General Flow (Ands used when multiple arguments are needed)

Given…

And…

When…

And…

Then…

And…

And…

And…

And…

1. We should have all three functions in place in a scenario for it to be considered a legitimate scenario.
   1. Per Recent updates from sessions with Mike, if we are only validating the content of the page, and we are starting on that page, then we do not need a when. We can exclude it in this situation.
2. One variable per line
3. Only call a variable once per scenario

# Best Practice for our scripting

1. We must focus on the feature at hand! Repetitive steps that get us to the feature we are scripting need to be included in the background as an assumption for the entire feature.
   1. In these cases, we need to be extra descriptive in the Background to make sure we aren’t creating any gaps for the automation team
   2. Ex: For the account summary feature file, logging in is not a step we should see in any scenario. Logging in should be included in the background
2. Avoiding repetitive scenarios across multiple feature files must be attained. To do this, we can say that any scenario who’s assertion results in another feature, should not be in the feature file.
   1. Exceptions here would be for critical path type features, like bill pay or enrollment where moving through different pages is going to be mandatory and within the same feature
   2. Ex: We should not have scenarios validating the header in the account summary feature file. These should only be in the Header Feature file
3. In terms of navigation, we don’t need to navigate to every target in the feature. We will focus on presence first, this let functionality and scope specific to the feature (see III-1) determine how to script the specific scenarios
4. We need to avoid validating any dynamic text. We can validate the functionality behind the field, but the text (if dynamic) is going to be t out of scope and in the hands of the LOB
   1. EX: BI box in IB2.
      1. Given I am on the account summary page
      2. When I go to inspect the BI box
      3. Then I should see the proper message in the BI box
5. Keep in general when creating scenarios that include elements like style, images or themes. These can be left out of the Gherkin Scripts, but need to be tracked somewhere so it is known that these cannot be automated
   1. Ex: The theme of the account summary between 12 am and 4:59 am
6. Every scenario in a Feature file should have an ID.

Ex: Add Payment Bill Pay for KMB Feature file, scenarios should have an ID, KMB\_Add Pay1, KMB\_AddPay2…

1. Do not use special characters including single and double quotes in Feature files
2. The name of feature file should be in lowercase and words separated by underscore (\_)
3. All features should be updated and added to sharepoint with extension .feature
4. No background will be used for the digital 17 project
5. Specific steps on navigation need not to be included like:
   1. Click on this button
   2. Select this button
   3. Click this