Assignment 1:

Case Study: Medication E-commerce Website - Treatment Journey Feature

Test Scenario:
Validation of the input fields of the Questionnaire:

Case 1:
Fields without values

- **Preconditions:** Customer can access the questionnaire page
- **Step**: Leave mandatory input field(s) empty and click the 'Submit' button
- **Test Data**:
  - Number of empty mandatory fields
    - 1
    - More than 1
    - All
  - Leave one or more optional fields empty
    - Yes
    - No
- **Expected Result:** It should follow the best practices & UX
  - _Option 1_*:* The submit button should remain disabled with a tooltip "Please select all mandatory input fields"
  - _Option 2_*:* An error message should appear indicating the missing info *(ideal practice)*

Case 2:
Fields with values, INVALID entries

- **Preconditions:** Customer can access the questionnaire page
- **Step**: Enter values according to Test data and click the 'Submit' button
- **Test Data**:
  - Numeric in free text fields
  - Text in numeric fields *(ideal practice, should be prevented)*
  - Email address
    - Incorrect format *(check examples here)*
    - Non-existent domain
  - HTML injection *(ideal practice, must be prevented)*
  - XSS injection *(ideal practice, must be prevented)*
  - SQL injection *(ideal practice, must be prevented)*
  - In the "Weight" input field
    - Enter value 0 *(should be prevented)*
    - Enter negative number *(should be prevented)*
  - In the "Date of Birth" input field
    - Enter incorrect format (by typing)
    - Enter a date < Min possible predefined value *(e.g., 31.12.1899) (should be prevented)*
    - Enter a date > Max possible value *(e.g., Today) (should be prevented)*

- **Expected result:** An error message should appear “Please check your input” *(ideally showing the appropriate type of error for the corresponding fields*

Case 3:
Fields with values, all VALID entries

- **Preconditions:** Customer can access the questionnaire page
- **Step**: Enter VALID values in all fields and click ‘Submit’ button
- **Test Data**:
    - Leave one or more optional fields empty:
        - No
        - Yes
- **Expected result**: Customer should redirect to the medicine dosage recommendation page

Case 4:
Fields with values, all VALID entries, customer is offline

- **Preconditions:** Customer can access the questionnaire page
- **Step**: Enter VALID values in all fields, disconnect the internet connectivity and click ‘Submit’ button
- **Expected result:** An error message should appear ‘Network Failure’ and the medicine dosage recommendation page should not load

Test Scenario:
Validation of the dose recommendation:

Case 1:
Customer without past medication dosage

- **Preconditions:** Customer can access the questionnaire page
- **Step**: Simulate a non-existent customer *(e.g., a unique email address)* while filling the form and click the ‘Submit’ button
- **Expected Result:** The medicine dosage page should appear and recommend a medicine with the <u>lowest</u> dose

Case 2:
Existing customer, non-existing medication dosage

- **Preconditions:** Customer can access the questionnaire page
- **Step**: Simulate an existing customer with <u>new set</u> of input data while filling the form and click the ‘Submit’ button
- **Test Data**:
    - Enter existing email address with <u>different</u>
        - First and/or Last name
        - DOB

        - Name(s) & DOB
        - *(Same customer may need medication for his/her family member(s))*
    - Enter existing email address with <u>different</u>
        - Symptom
        - *(Same customer may need medication for another disease)*
- **Expected Result:** The medicine dosage page should appear and recommend a medicine with the <u>lowest</u> dose

Case 3:

Existing customer, existing medication dosage, highest dose limit = not reached, preset time = not reached

- **Preconditions:** Customer can access the questionnaire page
- **Step**: While filling the form, simulate an <u>existing</u> customer
    - With <u>existing</u> "past medication"
    - The "dosage of the medication" has <u>not reached</u> its maximum limit
    - The "preset time" of the last dose has <u>not reached</u>

  Click the 'Submit' button

- **Expected Result:** The medicine dosage page should appear and recommend the medicine with the <u>same</u> dose as previous

Case 4:

Existing customer, existing medication dosage, highest dose limit = not reached, preset time = reached

- **Preconditions:** Customer can access the questionnaire page
- **Step**: While filling the form, simulate an <u>existing</u> customer
    - With <u>existing</u> "past medication"
    - The "dosage of the medication" has <u>not reached</u> its maximum limit
    - The "preset time" of the last dose has <u>reached</u>

  Click the 'Submit' button

- **Test Data:**
    - M = Number of days the last dose was provided
        - M = preset time
        - M > preset time
        - M >> preset time *(see Appendix 1)*
- **Expected Result:** The medicine dosage page should appear and recommend the medicine with the <u>next higher</u> dose than the current one

Case 5:

Existing customer, existing medication dosage, highest dose limit = reached

- **Preconditions:** Customer can access the questionnaire page
- **Step**: While filling the form, simulate an <u>existing</u> customer
    - With <u>existing</u> "past medication"

    - The “dosage of the medication” has reached its maximum limit

  Click the ‘Submit’ button

- **Test Data:**
    - M = Number of days the last dose was provided
        - M < preset time
        - M = preset time
        - M > preset time
- **Expected Result:** The medicine dosage page should appear and
    - *Option 1:* Recommend the medicine with the same dose as previous
    - *Option 2:* Show an info message about the highest possible dosage consumption based on the business model & medical compliances

Test Scenario:
Validation of the dose recommendation with same customer data:

Case:

Multiple customers with same sort of disease

- **Preconditions:** Customer can access the questionnaire page
- **Step**: While filling the form, simulate two customers having same disease credentials & click the ‘Submit’ button
- **Test Data:**
    - Use same disease related questions + same email address
        - Same time from different tabs/browsers
        - In different instances from same browser session
    - Use same disease related questions + different email addresses *(simulating different customers with same disease)*
    - Optionally use same personal details *(see Appendix 2)*
- **Expected Result:** The medicine dosage page should appear and recommend the same dose of same medicine for both

Appendices:

1. It is important to understand how the system behaves when a customer comes back to the platform after a long time. Should we stick to the same treatment progression rule or reset the history => I assumed the first one while writing test cases but it needs clarification from PO/PM
2. Ideally medicine doses have a dependency with the patients’ age and weight. If we want to consider them into the algorithm, it requires extra test case(s) covering different aged and weighing customers’ journey

Bug Investigation:

Title:
Wrong dose recommendation for highest dosage customers

Description:
When a customer has reached its maximum allowable dosage for a given medicine, the system recommends a lower level of dose.

Steps to reproduce:
Fill the questionnaire simulating an existing customer consuming the highest possible dosage for a given medicine after set period
Click the submit button
See the recommended dose

Expected Result:
The system will stick to recommend the same dose or show a message 'Customer has reached the highest level of dosage'.

Actual Result:
The system shows a lower dose than the current one

Attachments:
Add all necessary screenshots

Severity Level:
High

Fix Version:
To be decided

Components/Label:
User Journey, Recommendation

Compliance Testing:

In order to ensure the legal and medical compliances of the system, we should proceed as follows:

- Make sure the policy pages (Privacy, Fair use, Cookie, T&C) are well visible and accessible for the users, both in desktop & mobile (browser + App)
- All these pages contain correct information including last update
- Preferably while accessing in Web, these pages open in a new browser tab
- If the webpage supports multiple interface languages, all these pages
    - are available for all supported languages
    - follow a default language as a fallback
- While filling up the questionnaire, customers must accept the legal T&C and consent their personal data handling for GDPR reason
- For certain medicines that are strictly protected for minors, should not be recommended to a customer having age <18
- For certain medicines that entail different doses for adults & minors, the correct dose should be recommended to the ideal age group

- The recommendation should obey the maximum dose limit for all customers, irrespective to the medicine type

Process Improvements:

To improve the quality of the recommendation feature and the E2E testing process, it is necessary to document clear & concise feature description, especially defining the boundary conditions and technical acceptance criteria. If I need some clarifications regarding the questionnaire logic, I will move forward with the followings as early as possible *(shift-left)*:

1. Note down & accumulating the doubts and question I have after going through the feature description *(see sample NOTES below)*
2. Schedule a meeting with the PO to clarify about them
3. Put my points of concerns and if possible, suggest alternative possibilities
4. Involve the responsible developer(s) to
    a. Understand the bottleneck(s)
    b. ensure the feasibility of all possible alternatives/improvement proposals
5. If no changes/updates seem to be feasible, we should come up with a mitigation plan
6. In the end, all the stakeholders should conclude the outcome by:
    a. **PO:** Updating the feature description (logic, flow-charts, design patterns etc.)
    b. **QA:** Updating the test cases (if needed)
    c. **Dev:** Modifying/Reworking the algorithm
7. Irrespective to the discussion result (positive/negative), this should be a part of next retrospective meeting

MY sample NOTES:

1. What if I want to purchase the same dosage as before, even after 1 month?
2. Is there any possibility to skip the questionnaire?
3. Pre-filled values (e.g., Personal details) while entering email address (fetch from DB)
4. Adding family members to avoid creating >1 user account
5. Compliance => Age verification (e.g., Medicine strictly applicable for 18+)
6. Recommended dose while upper bound has reached?
7. How to handle the recommendation if the medicine is out of stock?
    a. Simply display it is not available
    b. Recommend an alternative medicine
        i. Obey the continuation of past dosage metric?
        ii. Start from the lowest level?

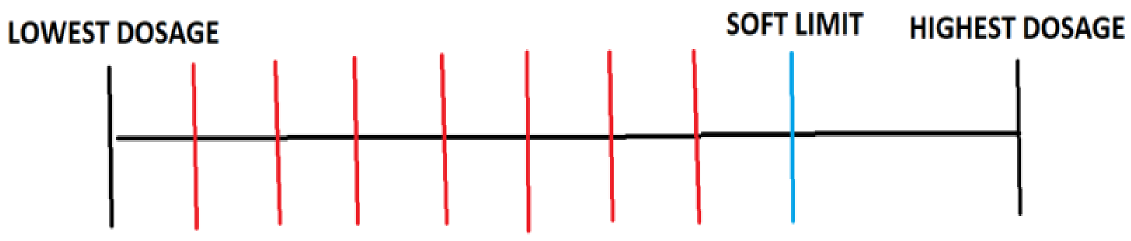
Collaboration:

I also agree this is a critical issue that needs to be resolved before production. But we should not delay the release due to its fix. Hence, we need a proper mitigation plan that should

- Put down the compliance risk factor
- Avoid any revenue loss

To ensure things to be done, I would like to come up with multiple intermediate proposals within the scope of a mitigation discussion. For example,

- Increase the intermittent set period so that the probability of reaching the threshold lowered down further.
- Show an info message upon reaching the highest dose that you're already consuming a higher dose & they should agree to keep the same dose for the moment
- Impose a soft limit of each medicine dosage and make the set period "infinite" between soft and highest limit so that customer would never reach the highest limit



We need to make sure to conclude one solution with everyone's consent. Once the intermittent solution imposed, we have to create a follow up ticket and add it into the next upcoming release.

Assignment 2

Case Study: Automation Testing for E-commerce Website – Checkout Page and Coupon Code Feature

Test Scenario:

Validation of coupons in the checkout section:

Case 1:
EMPTY Coupon

- **Preconditions:** Desired product has selected; credentials are filled successfully
- **Step**: Click the 'Gutschein hinzufügen' => Keep the input field blank and click 'Hinzufügen'
- **Expected Result:**
    - An error message should appear 'Der Gutschein "" konnte leider nicht zu deiner Bestellung hinzugefügt werden, da er ungültig ist.'
    - The total sum should be 476,00 €

Case 2:
INVALID Coupons

- **Preconditions:** Desired product has selected; credentials are filled successfully
- **Test Data:**
    - INVALID Coupon Code
    - EXPIRED VALID Coupon
    - MULTIPLE VALID Coupon
- **Step**: Click the 'Gutschein hinzufügen' => Enter coupon code as per Test Data => Click 'Hinzufügen'
- **Expected Result:**
    - An error message should appear 'Der Gutschein <*"entered value"*> konnte leider nicht zu deiner Bestellung hinzugefügt werden, da er ungültig ist.'
    - The total sum should be 476,00 €

Case 3:
VALID Coupons

- **Preconditions:** Desired product has selected; credentials are filled successfully
- **Test Data:**
    - Enter coupon by
        - Pasting
        - Keyboard typing
    - Not expired valid Coupon with
        - all lower case
        - all upper case
        - combination of above

- **Step**: Click the ‘Gutschein hinzufügen’ => Enter coupon code as per Test Data => Click ‘Hinzufügen’
- **Expected Result:**
    - An confirmation message should appear ‘Der Gutschein wurde erfolgreich zu deiner Bestellung hinzugefügt’
    - The total sum will reduce to 466,00 €

Case 4:
USED Coupon

- **Preconditions:** Desired product has selected; credentials are filled successfully
- **Step**: Click the ‘Gutschein hinzufügen’ => Enter an used coupon => Click ‘Hinzufügen’
- **Expected Result:**
    - An info message should appear ‘Der Gutschein wurde bereits verwendet. Bitte legen Sie ein neues ein. ***(Assumption)***
    - The total sum should be 476,00 €

Handling edge cases in Automation

Handling edge cases in automation depends on the type of edge case under consideration.

Test Scenario 1:

If the automation is designed only for regression testing, then it is not a common practice to include edge cases within automation coverage. Usually, they are covered by manual execution and part of in-depth feature testing.

Test Scenario 2:

If the automation is a part of E2E testing, then edge cases should be a part of that as well. In principle, it should be checked “separately” from the normal E2E user flows. It helps to identify issues quickly and retest after a fix *(if applicable).* Once the fix is done, you don’t need to run the whole batch of test cases but only the affected one.

Data-Driven Testing

Data-driven testing is an approach where test data are separated from test scripts and stored in external files, such as JSON, CSV, Excel, TXT. This method allows for the same test to be run multiple times using different sets of data.

In **Cypress**, data-driven testing is typically accomplished using the fixture command. Fixtures is used to handle external data files and can be easily loaded within tests. It

- reduces code complexity
- enhances encapsulation by keeping the test data independent from the test scripts
- simplifies test management
- enhances input handling

In the current COUPON code testing scenario, I have created a .json file to store all the applicable codes. Using the ***cy.fixture*** command, all coupons are directly loaded within the tests and utilized across various scenarios.

For **CSV files**. we need to follow certain steps:

1. install ***neat-csv package*** within Cypress***.*** It should be available in ***package. json*** file
2. Add the ***.csv*** file into the ***fixture folder***
3. Import ***neat-CSV*** library as ***import neatCSV = require('neat-csv')***
4. Follow the same procedure like ***cy.fixture*** command
5. In the end, the parsed data will store into a ***variable*** which will use for different scenarios
6. It is always available to see the parsed file by writing the command as ***console.log(variableName)***

Example how we use the parsed csv data:

cy.get(css selector).type(variableName[index][validCoupon /expiredCoupon/maxUseRechCoupon])

For **EXCEL files**. we need to follow these steps:

1. Install ***xlsx package*** within Cypress
2. Define the task that reads and parses the Excel file and make changes accordingly to the ***cypress/plugins/index.js***
3. Now it is ready to invoke the task into the ***cypress*** test scripts. It is always available to see the parsed file by writing the command as ***console.log(variableName)***

Collaboration and Reporting:

If such issue is identified during testing, I would like to go through the following granular steps:

1. I will try to debug & check the error log in Cypress
2. Try to reproduce the same issue via manual testing
3. If still reproducible, as part of the root cause analysis, I will try to reproduce the same issue using another valid coupon having same category (e.g., having same entities in the DB)
4. After collecting facts and data, I will raise a bug reporting the exact class of issue, supported by screenshots and enough data to reproduce
5. Once it is fixed, I will try to reproduce it again manually
6. If it is working, I will rerun the automation script and try to reproduce

To reduce the testing gaps, it is important to prepare handful of test data before & after the fix. This helps to reproduce the issue seamlessly and keep the testing coherence.