**QA Take Home**

*Pretend you are new to a company and in the process of onboarding. You are assigned to your first piece of functionality and are asked to create a written test plan, as well as, draft automation for drop-downs across the application. Below you will find product requirements and mocks, to provide guidance for the written test plan, and background knowledge that a fellow QA engineer has provided you to help with your automation.*

**Product Requirements:**

A user has two sources of truth – their own crm data, and data from an external library. They need to be able to find matches across the two sources. The more information they provide upfront the better the quality of the matches will be, but only some fields/query parameters are required to produce a match.

A dropdown with the following

* Ability to (multi-) select the categor(ies) (listed below) to use for finding matches against a user’s CRM fields. Fields required for mapping within each section should be displayed on selection of the category.
* If an input is required for multiple selected categories, it should only be displayed and mapped once.
* Deselection of a category can be done by un-selecting the applicable option within the dropdown. Fields within this category are then removed from the list, except for fields being used by another category selected at the time.
* Once a category is selected, the required field mappings display within the table:

|  |  |  |  |
| --- | --- | --- | --- |
| Category | Query Parameters | Required | Optional |
| ID | SSN | X |  |
| Personal Info | First Name | X |  |
| Last Name | X |  |
| Address | X |  |
| Education Level |  | X |
| Age |  | X |
| General Info | First Name | X |  |
| Last Name | X |  |
| Job | X |  |
| Favorite Food |  | X |
| Favorite Book |  | X |
| Contact Info | phone | OR |  |
| email |  |

* Ability to select the source CRM field(s) that will be used to identify matches.
* Ability to progress to the next step of the module when at least one category has all required fields mapped.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

**Instructions (Part I):**

Write test cases for this feature. If something is unclear, something that you would typically clarify with product or engineering before finishing your test cases please add those questions at the bottom of your test cases.

**Behind the scenes QA knowledge:**

**Dropdown Variations:**

* Some trigger network calls, show spinners, or have delayed rendering.
* Some are searchable and others are not
* There are multi-select and single-select variations.
* Some show selected values as badges, others show them as inline text or trigger a behavior.
* Some dropdown selections enable other fields or buttons, so the utility needs to wait for UI changes or DOM updates.

**Instructions (Part 2):**

Based on your “behind the scenes knowledge” of the drop-downs in the product, write modular/reusable automated tests for dropdowns across the application within the following guidelines (use whatever syntax/pseudo-syntax you feel most comfortable with, but please identify it and if you know Cypress, please use that):

**Must-follow Constraints:**

* No force: true allowed — must resolve visibility or state issues correctly.
* Script length should not exceed ~75 lines. Keep it readable and focused.
* Use dynamic waiting techniques — avoid hardcoded wait() calls.
* Should wait for:
  + Network requests to complete
  + DOM changes after interactions
  + UI render delays (e.g., dropdown open/render, option visible)
  + Should operate like a real user would, not like a robot.
  + Avoid immediately chaining actions — the flow should mimic intentional, human-paced behavior.
* Code should be modular and reusable, not just a fixed flow.
* External packages are allowed.
* No flake under network error.
* Proper error handling and show custom error message.