Testing Document

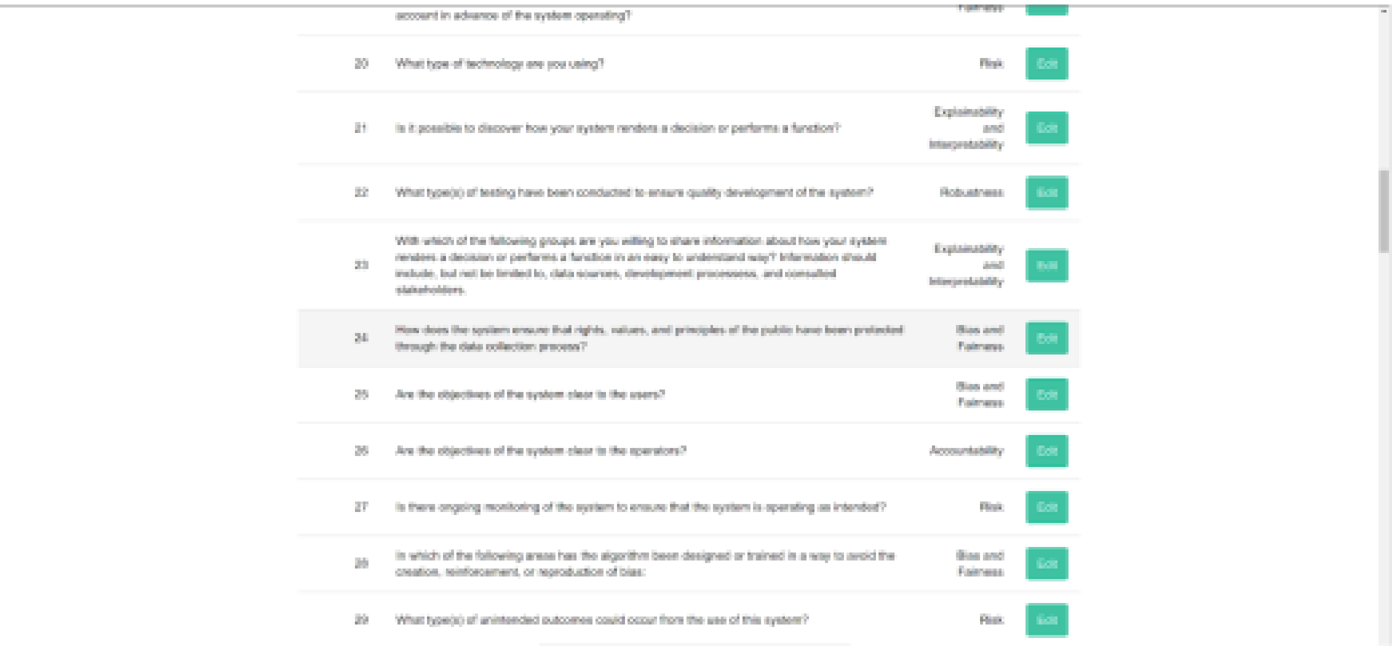
November 23, 2020
Dalton Ronan, Harrison Peters

Status: Not updated

Manual Tests

Reordering Survey Questions

Test if an admin user can re-order survey questions by dragging a row in the table to a new location. It covers an aspect of the admin question management functionality **(US 5.07)**

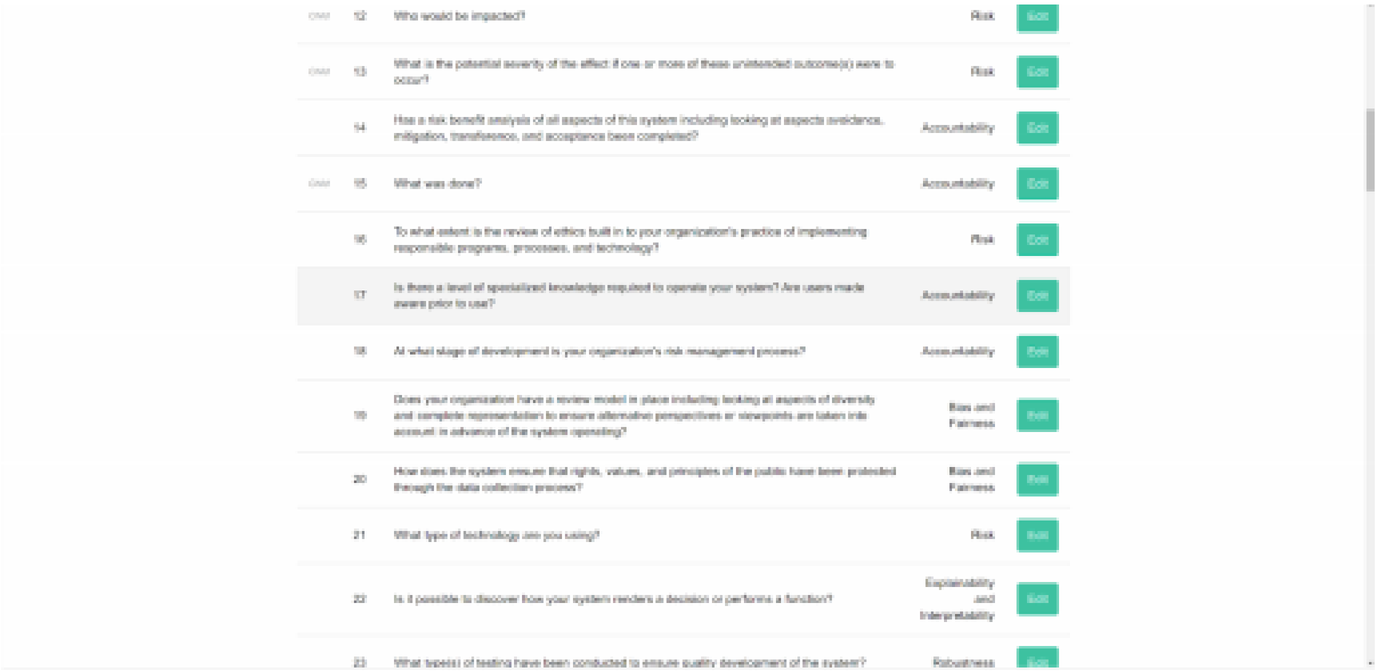


	account in advance of the system operating?	Fairness	Edit
20	What type of technology are you using?	Risk	Edit
21	Is it possible to discover how your system renders a decision or performs a function?	Explainability and Interpretability	Edit
22	What type(s) of testing have been conducted to ensure quality development of the system?	Robustness	Edit
23	With which of the following groups are you willing to share information about how your system renders a decision or performs a function in an easy to understand way? Information should include, but not be limited to, data sources, development processes, and consulted stakeholders.	Explainability and Interpretability	Edit
24	How does the system ensure that rights, values, and principles of the public have been protected through the data collection process?	Bias and Fairness	Edit
25	Are the objectives of the system clear to the users?	Bias and Fairness	Edit
26	Are the objectives of the system clear to the operators?	Accountability	Edit
27	Is there ongoing monitoring of the system to ensure that the system is operating as intended?	Risk	Edit
28	In which of the following areas has the algorithm been designed or trained in a way to avoid the creation, reinforcement, or reproduction of bias:	Bias and Fairness	Edit
29	What type(s) of unintended outcomes could occur from the use of this system?	Risk	Edit

1. Select a question to re-order by clicking and holding the `left-mouse` button, or by using the `tab` key to navigate to a row, and press the `space key` to select.

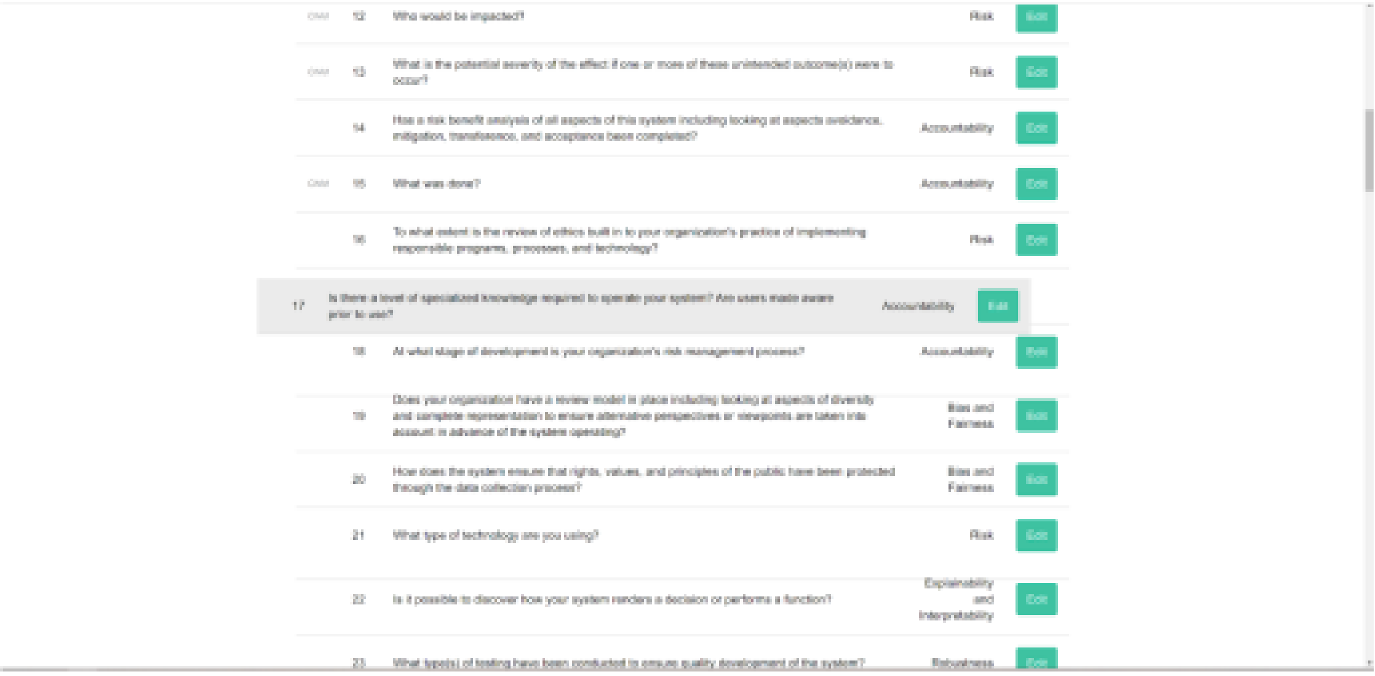
Creating Hierarchy Questions

Test is an admin user can create hierarchy question relationships from the admin panel, and that the hierarchy questions are presented to the survey user **(US 5.02)**.



Child	12	Who would be impacted?	Risk	Edit
Child	13	What is the potential severity of the effect if one or more of these unintended outcome(s) were to occur?	Risk	Edit
	14	Has a risk benefit analysis of all aspects of this system including looking at aspects avoidance, mitigation, transference, and acceptance been completed?	Accountability	Edit
Child	15	What was done?	Accountability	Edit
	16	To what extent is the review of ethics built in to your organization's practice of implementing responsible programs, processes, and technology?	Risk	Edit
	17	Is there a level of specialized knowledge required to operate your system? Are users made aware prior to use?	Accountability	Edit
	18	At what stage of development is your organization's risk management process?	Accountability	Edit
	19	Does your organization have a review model in place including looking at aspects of diversity and complete representation to ensure alternative perspectives or viewpoints are taken into account in advance of the system operating?	Bias and Fairness	Edit
	20	How does the system ensure that rights, values, and principles of the public have been protected through the data collection process?	Bias and Fairness	Edit
	21	What type of technology are you using?	Risk	Edit
	22	Is it possible to discover how your system renders a decision or performs a function?	Explainability and Interpretability	Edit
	23	What type(s) of testing have been conducted to ensure quality development of the system?	Robustness	Edit

1. Select a question to re-order by clicking and holding the `left-mouse` button, or by using the `tab` key to navigate to a row, and press the `space key` to select.

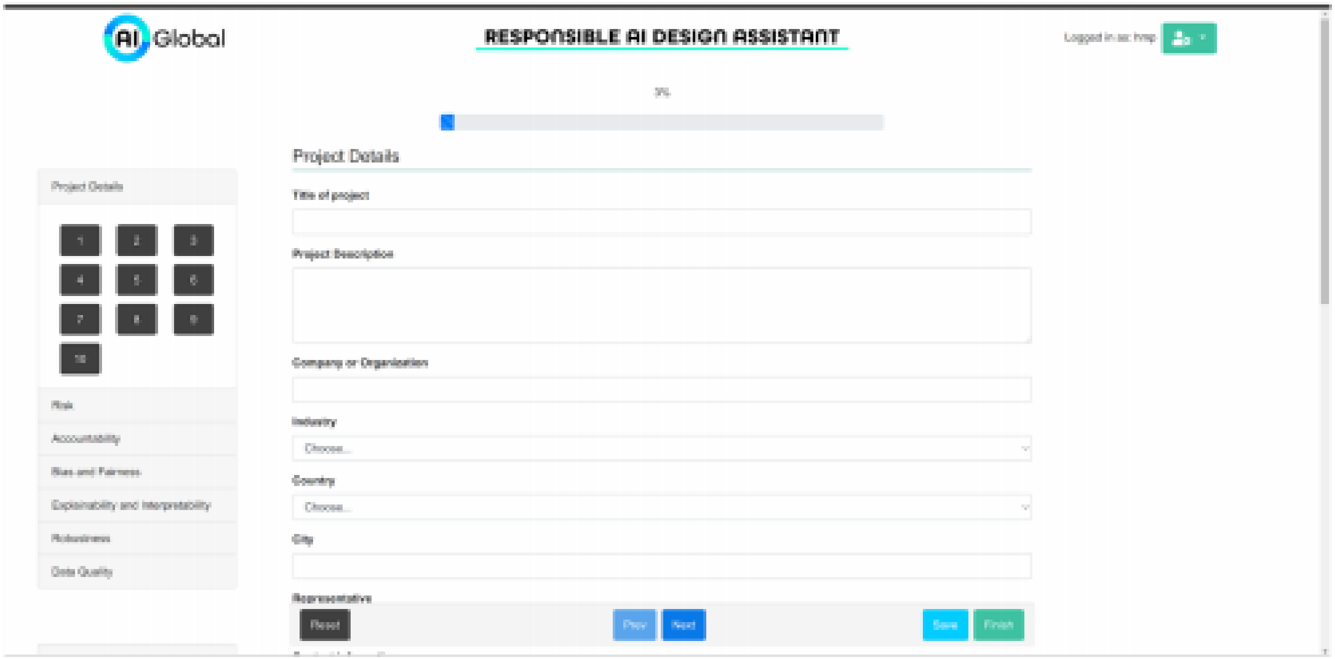


[illegible]	12	Who would be impacted?	Risk	Edit
[illegible]	13	What is the potential severity of the effect if one or more of these unintended outcome(s) were to occur?	Risk	Edit
	14	Has a risk benefit analysis of all aspects of this system including looking at aspects avoidance, mitigation, transference, and acceptance been completed?	Accountability	Edit
[illegible]	15	What was done?	Accountability	Edit
	16	To what extent is the review of ethics built in to your organization's practice of implementing responsible programs, processes, and technology?	Risk	Edit
	17	Is there a level of specialized knowledge required to operate your system? Are users made aware prior to use?	Accountability	Edit
	18	At what stage of development is your organization's risk management process?	Accountability	Edit
	19	Does your organization have a review model in place including looking at aspects of diversity and complete representation to ensure alternative perspectives or viewpoints are taken into account in advance of the system operating?	Bias and Fairness	Edit
	20	How does the system ensure that rights, values, and principles of the public have been protected through the data collection process?	Bias and Fairness	Edit
	21	What type of technology are you using?	Risk	Edit
	22	Is it possible to discover how your system renders a decision or performs a function?	Explainability and Interpretability	Edit
	23	What type(s) of testing have been conducted to ensure quality development of the system?	Robustness	Edit

2. Move the question to the desired position by dragging it with the mouse and releasing the button (if held and clicked in in step 1), or the pressing the `arrow keys` and pressing `space` again (if used the `space key` in step 1).

Survey Navigation and Question Markers

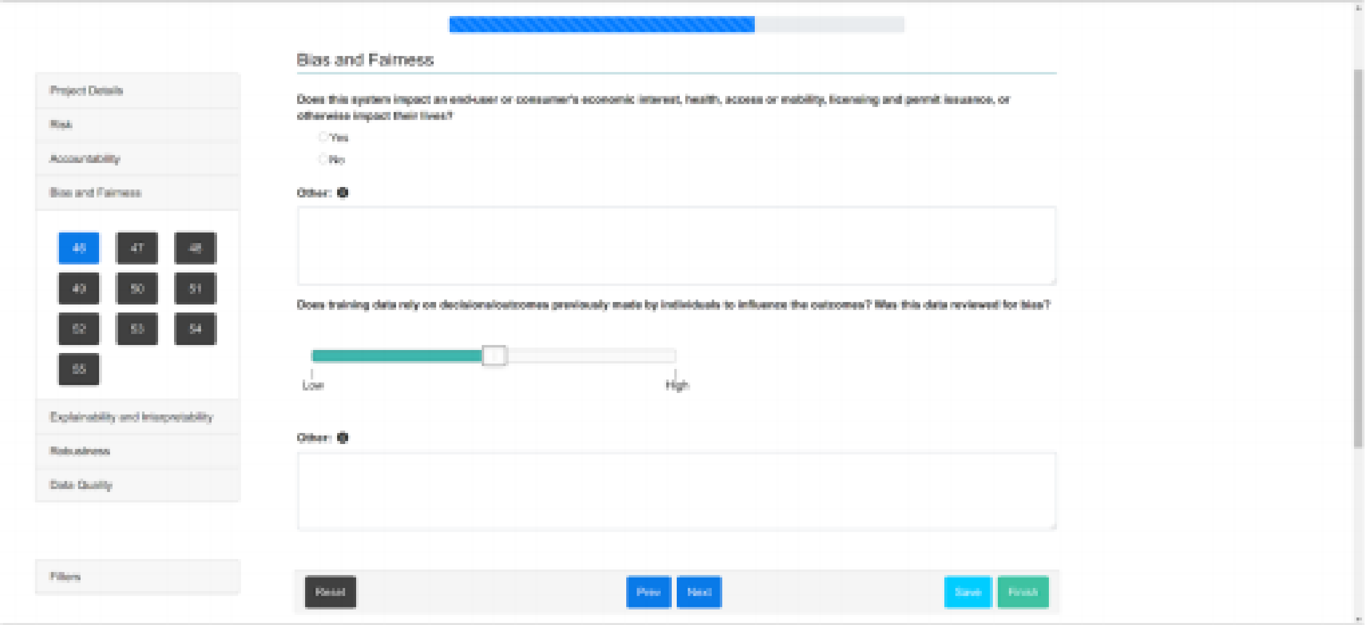
Tests that the user has navigation controls that lets them easily navigate between survey dimensions and questions. The navigation cards display the questions as grey when they are unanswered, and turn blue as they are answered **(US 1.01, US 1.02)**.



1. Survey starts with navigation controls to each of the question navigations.

Spectrum Questions

Test that users have spectrum questions instead of check boxes for certain questions **(US 1.04)**.



Bias and Fairness

Does this system impact an end-user or consumer's economic interest, health, access or mobility, licensing and permit issuance, or otherwise impact their lives?

○ Yes

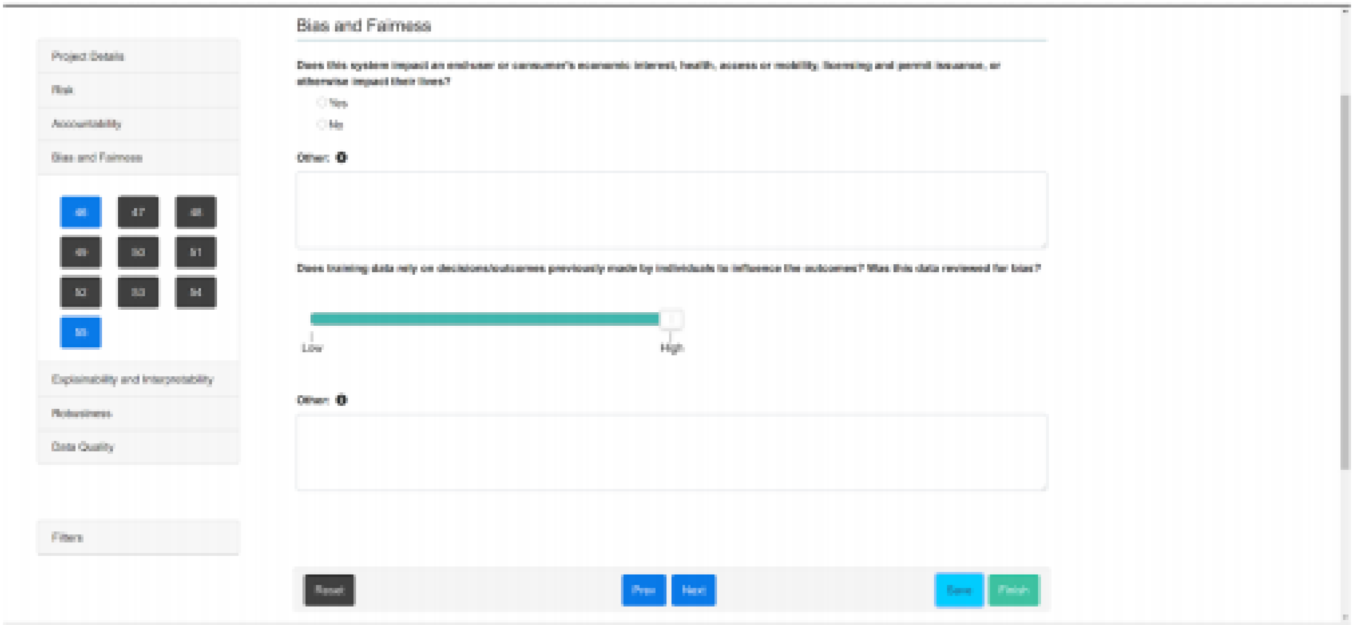
○ No

Other:

Does training data rely on decisions/outcomes previously made by individuals to influence the outcomes? Was this data reviewed for bias?

Other:

1. Certain questions are presented as a slider.



Bias and Fairness

Does this system impact an end-user or consumer's economic interest, health, access or mobility, licensing and permit issuance, or otherwise impact their lives?

○ Yes
○ No

Other:

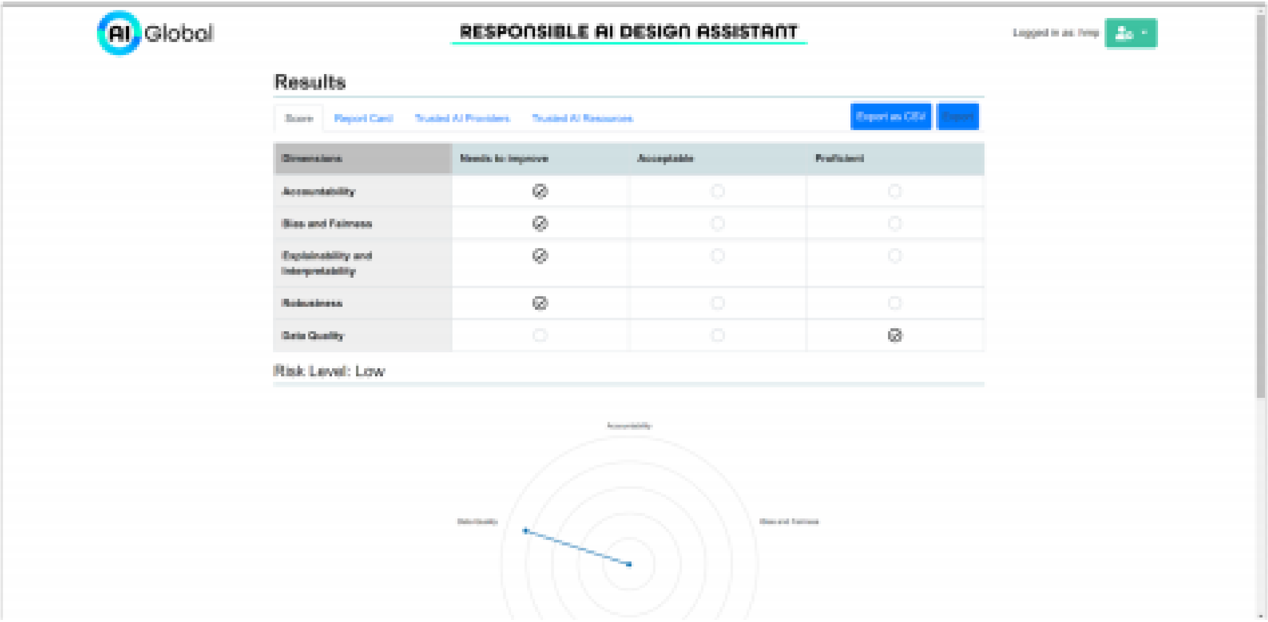
Does training data rely on decisions/outcomes previously made by individuals to influence the outcomes? Was this data reviewed for bias?

Other:

2. The use can click and slide it between the *Low* and *High*, to the point that best represents their response.

Report Card Title and Description

Test that the exported PDF report card contains the new Title and Description **(US 4.04)**.

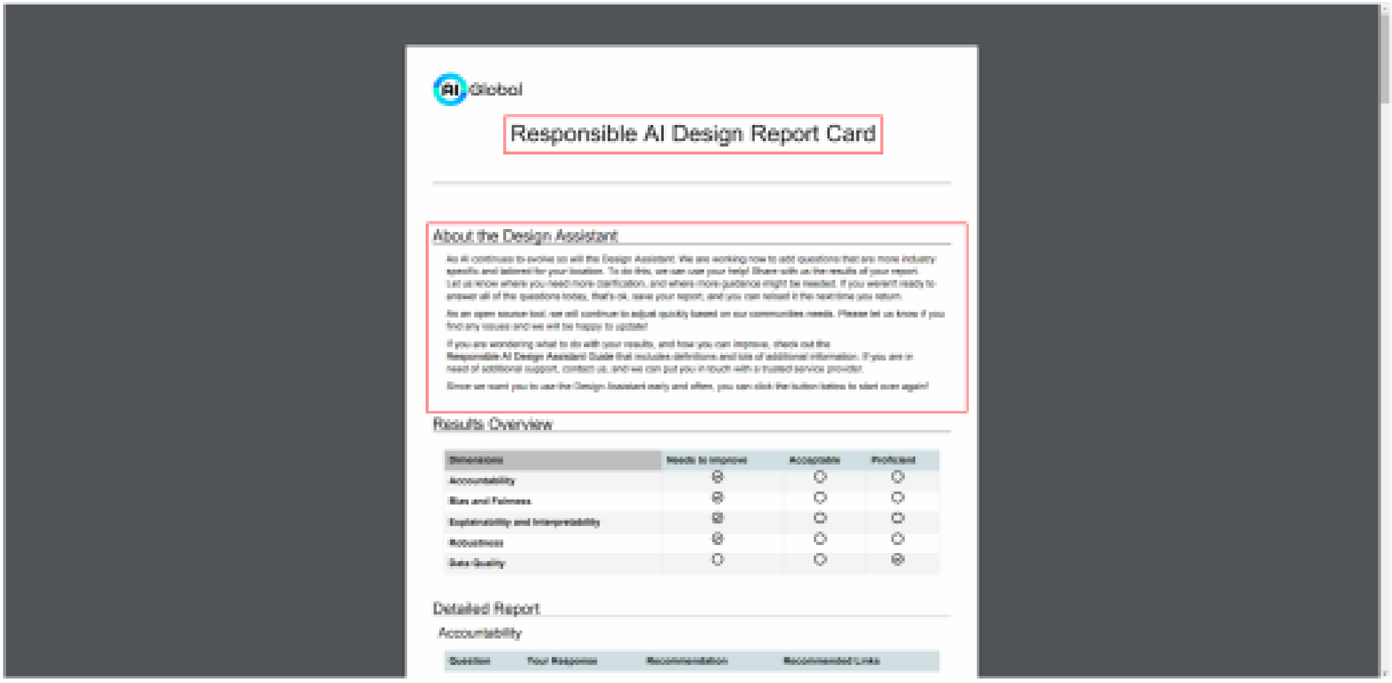


Results

Risk Level: Low

Dimensions	Needs to improve	Acceptable	Proficient
Accountability	☒	○	○
Bias and Fairness	☒	○	○
Explainability and Interpretability	☒	○	○
Robustness	☒	○	○
Data Quality	○	○	☒

1. From the results page, a user can click the *Export* button to export the detailed report card as a pdf.



AI Global

Responsible AI Design Report Card

About the Design Assistant

As AI continues to evolve so will the Design Assistant. We are working now to add questions that are more industry specific and tailored for your location. To do this, we can use your help! Share with us the results of your report. Let us know where you need more clarification, and where more guidance might be needed. If you weren't ready to answer all of the questions today, that's ok, save your report, and you can reload it the next time you return.

As an open source tool, we will continue to adjust quickly based on our communities needs. Please let us know if you find any issues and we will be happy to update!

If you are wondering what to do with your results, and how you can improve, check out the Responsible AI Design Assistant Guide that includes definitions and lots of additional information. If you are in need of additional support, contact us, and we can put you in touch with a trusted service provider.

Since we want you to use the Design Assistant early and often, you can click the button below to start over again!

Results Overview

Dimensions	Needs to improve	Acceptable	Proficient
Accountability	●	○	○
Bias and Fairness	●	○	○
Explainability and Interpretability	●	○	○
Robustness	●	○	○
Data Quality	○	○	●

Detailed Report

Accountability

Question	Your Response	Recommendation	Recommended Links

2. The exported pdf displays a title and the Design Assistant description at the top.

Automated Tests

All automated React unit tests are found in the `frontend/src/tests/unit_tests` project directory.

Admin.test.js

This suite tests if the admin page successfully renders for an admin user to make sure all admin User Stories **(US 5.04, US 5.06)** will be accessible to an admin.

Admin Page Renders:
Tests if the administration panel page successfully renders with a mock empty database. Checks for expected UI elements on successful render.

Admin Page renders survey management:
Tests if the administration panel page successfully renders with a mock database with question data. Checks for expected UI elements on successful render and correct presentation of question data in table.

Admin Page renders submission management:
Tests if the administration panel page successfully renders and has a tab to view User Submissions. Clicks the *Submissions* tab to confirm it is navigable.

Admin Page renders analytics tab:
Tests if the administration panel page successfully renders and has a tab to view

User Submissions. Clicks the *Analytics* tab to confirm it is navigable.

AdminProviders.test.js

This suite tests the functionality for an admin user to be able to add, edit and remove Trusted AI Providers that can be displayed to a user completing the survey **(US 5.09)**.

Admin Trusted AI Providers renders successfully:
Tests if Trusted AI Providers component of Admin Panel renders successfully with a list of Trusted AI Providers.

Admin Trusted AI Providers renders successfully with no data:
Tests if Trusted AI Providers component of Admin Panel renders successfully with no data.

Admin Trusted AI Providers create new button opens edit modal:
Tests if Admin can successfully add a new AI Provider by opening a modal by clicking the *'add provider'* button.

Admin Trusted AI Providers edit button opens edit modal:
Tests if Admin can successfully edit an existing AI Provider by opening a modal by clicking the *'Edit'* button associated with the AI Provider table entry.

Admin Trusted AI Providers edit modal successfully submits new provider:
Tests if a new AI provider can be successfully added to the database through the admin panel. It opens the modal by clicking the *'add provider'* button, enters data into the input fields, saves the new Provider by clicking the *'save'* button, and confirms the list of AI Providers contains the new entry.

Admin Trusted AI Providers edit modal fails source validation:
Tests that the Admin AI Provider function prevents the admin from adding an already existing AI provider to the database. It opens the modal by clicking the *'add provider'* button, enters data for an AI Provider that is already found in the database, and checks that the failed validation message is displayed when the *'save'* button is clicked.

Admin Trusted AI Providers edit modal successfully updates provider: Tests that editing an existing AI Provider in the admin panel successfully updates the information stored in the database. It opens the modal by clicking the *'Edit'* button, enters new data for the AI Provider, clicks the *'save'* button, and confirms the changes are reflected in the new list.

AdminResources.test.js

This suite tests the functionality for an admin user to be able to add, edit and remove Trusted AI Resources that can be displayed to a user completing the survey **(US 5.09)**.

Admin Trusted AI Resources renders successfully:
Tests if Trusted AI Resources component of Admin Panel renders successfully with a list of Trusted AI Resources.

Admin Trusted AI Resources renders successfully with no data:

Tests if Admin can successfully add a new AI Resource by opening a modal by clicking the *'add resource'* button.

Admin Trusted AI Resources create new button opens edit modal:
Tests if Admin can successfully add a new AI Resource by opening a modal by clicking the *'add resource'* button.

Admin Trusted AI Resources edit button opens edit modal:
Tests if Admin can successfully edit an existing AI Resource by opening a modal by clicking the *'Edit'* button associated with the AI Resources table entry.

Admin Trusted AI Resources edit modal successfully submits new resource: Tests if a new AI Resource can be successfully added to the database through the admin panel. It opens the modal by clicking the *'add resource'* button, enters data into the input fields, saves the new Provider by clicking the *'save'* button, and confirms the list of AI Resources contains the new entry.

Admin Trusted AI Resources edit modal fails source validation:
Tests that the Admin AI Resources function prevents the admin from adding an already existing AI Resource to the database. It opens the modal by clicking the *'add resource'* button, enters data for an AI Resources that is already found in the database, and checks that the failed validation message is displayed when the *'save'* button is clicked.

Admin Trusted AI Resources edit modal successfully updates resource: Tests that editing an existing AI Resources in the admin panel successfully updates the information stored in the database. It opens the modal by clicking the *'Edit'* button, enters new data for the AI Resources, clicks the *'save'* button, and confirms the changes are reflected in the new list.

App.test.js

This suite checks if the Responsible AI Design Assistant Survey Welcome page is successfully rendered. It ensures that the App and its features are accessible. From here, users can log in, sign up, complete a survey, access previous surveys, etc. and checks that a non-logged in user can still complete the survey **(US 1.05)**

Welcome Page renders:
Tests that the welcome page and text successfully renders and the login/signup button is accessible.

Welcome Page renders with no data:
Tests that the welcome page and text successfully renders and the login/signup button is accessible.

DesignAssistantSurvey.test.js

This suite tests some of the main functionalities and features of the Responsible AI Design Assistant survey are rendered and accessible to the user **(US 1.01, US 1.02, US 1.03, US 1.04, US 1.05, US 2.01, US 2.02, US 2.03)**.

Survey Renders:

Tests that the survey questions JSON is successfully loaded into the SurveyJS Model and displayed to the user, starting on Page 1.

Survey Page reset button pops modal to return to Home page:
Tests that the survey can be reset at any time by pressing the *'Reset'* button, which triggers a modal to ask the user to confirm they want to reset the survey.

Survey Page Next button renders page 2:
Tests that the survey can navigate to the next page by clicking the *'Next'* button, and confirming the page is displaying Page 2. Also tests that the *'Prev'* button navigates back to Page 1.

Survey Page finish button submits the survey:
Tests that the survey can be finished by clicking the *'Finish'* button.

Survey Page can open accordion to navigate dimensions:
Tests that the Navigation accordion menu renders beside the main page content, and that the menu headers depend on the Dimensions received from the API call, and the headers a clickable to dropdown the card containing that dimensions question, so the user has the ability to navigate the survey via this menu.

Survey Page can open accordion to filter select roles:
Tests the the Filters accordion menu renders beside tha main page content, and that dropdown menus for available filters are selectable, and can be applied by clicking the *'Apply Filters'* button.

DimensionScore.test.js

This suite tests the Dimension Score section of the final report that the user is presented upon completion of the survey.

Dimension Score renders:
Tests that the page successfully renders the data for a dimension.

Dimension Score renders with no data:
Tests that the page successfully renders with no data.

Dimension Score marks "Needs to improve":
Tests that the page successfully renders the data for a dimension, and that it is marked as *'Needs to improve'* with a score of 1.

Dimension Score marks "Acceptable":
Tests that the page successfully renders the data for a dimension, and that it is marked as *'Acceptable'* with a score of 2.

Dimension Score marks "Proficient":
Tests that the page successfully renders the data for a dimension, and that it is marked as *'Proficient'* with a score of 4.

Login.test.js

This suite tests the login functionality for the App, and that a user can begin the process of creating an account if they do not have one **(US 3.01, US 3.02)**.

Log in button renders successfully with no user logged in:
Tests that the login button renders when a user is not logged in.

Logged in username renders successfully with user logged in:
Tests that the login button renders and the page displays the users username when the user is logged in.

Log in button successfully transitions to login page:
Tests that clicking the *'login'* button brings up the login modal for the user to enter their login information.

Login page fails username validations:
Tests that entering the wrong username will not log the user in and will provide them with feedback: *wrong username*.
Login page fails password validations:
Tests that entering the wrong username will not log the user in and will provide them with feedback: *wrong password*.

Login page successfully logs user in:
Tests that entering a correct username and password combination will log the user in and validate a login token.

Create an account button successfully transitions to signup page:
Tests that the user can select the *'Create your account'* option from the login modal to begin the signup process.

QuestionModal.test.js

This suite test the functionality of the edit question modal, which gives the admin an interface to add, edit and remove questions in the database **(US 5.07)**

Question Modal renders with Project Details (Tombstone) question:
Tests that the edit question modal renders properly for editing Project Details/Tombstone questions. It expects the following fields to not have rendered, since they are not needed for these types of questions: Weight, Role, Responses, Score, Reference.

Question Modal renders with survey Risk question:
Tests that the edit question modal renders properly for editing risk (equivalently, mitigation) questions. It expects all question fields to render in and be editable.

Question Modal renders with new question:
Tests that the edit question modal renders properly for adding new questions. It expects Weight, Role, Responses, Score, Reference to not render on open. It tests that trying to save a question without the question field filled in triggers a validation warning: *please enter a question*, and does not close the modal. Tests that using the dropdown menu to select a different question type (risk) successfully renders in the new necessary fields.

Question Modal renders with a survey Mitigation question:
Tests that the edit question modal renders properly for editing risk (equivalently,

mitigation) questions. It expects all question fields to render in and be editable. Checks that Reference, Alt Text, and Link fields correctly render the values from the database.

Question Modal renders with survey slider question and does not save invalid data : Tests that the edit question modal renders properly for editing slider questions. It checks the *Low*, *Med*, and *High* fields render. It fires the save button with these fields and the question field empty, and confirms that validation warnings: *invalid* for the slider points*, Please enter a question* for the question field, are displayed, since these are required fields.

Question Modal can delete a question:
Tests that a question can be deleted from the edit modal by clicking the *'Delete'* button. When the *'Delete'* button is clicked, confirms that a warning is displayed to the admin, and they can select *'Cancel'* to return to editing mode, or *'Yes'* to delete the question from the database.

ReportCard.test.js

This suite tests that the Report Card page successfully renders for a user on completion of a survey.

Report Card renders:
Tests that the report card renders survey results and dimension data.

Report Card renders with no data:
Tests that the report card renders survey results and dimension data.

Results.test.js

This suite tests that the Results page renders and displays all the necessary information to the user after they have completed the survey, including the Dimension Score, Report Card, Trusted AI Providers, and Trusted AI Resources **(US 4.03, US 4.04)**.

Results successfully renders:
Tests that the results page renders with completed survey data.

Results renders with no data:
Tests that the results page renders with completed survey no data.

Results redirects home if survey incomplete:
Tests that the survey results page redirects the user to the Welcome page if the survey is not completed.

Results start again button does not error out:
Tests that the *Start Again'* button can be clicked to return the user to the Welcome page to begin a new survey.

Results switches to Report Card Tab:
Tests that the user can navigate to the Report Card page by clicking the table tabs, so they can view a detailed breakdown of their responses.

Results switches to Trusted AI Providers Tab:
Tests that the user can navigate to the Trusted AI Providers tab by clicking on the corresponding tab so they can view the provided resources set up by the Admin.

Results switches to Trusted AI Resources Tab:
Tests that the user can navigate to the Trusted AI Resources tab by clicking on the corresponding tab so they can view the provided resources set up by the Admin.

Signup.test.js

This suite tests the signup functionality of the App, which allows the user to create an account so they can have their responses saved and can complete and track multiple assessments **(US 3.01)**.

Signup renders successfully:
Tests that the sign up interface successfully renders so the user can enter their desired account information.

Signup fails email validation:
Tests that a user cannot create a new account with an email that is associated with an existing account by entering an existing email address, checking for a validation warning: *email already exists*, and not completing the sign up process.

Signup fails password validation:
Tests that a user cannot create a new account with an invalid password by entering a password that does not satisfy the strength criteria, checking for a validation warning: *weak password*, and not completing the sign up process.

Signup fails confirm password validation:
Tests that a user cannot create a new account with an invalid password by entering passwords that do not match, checking for a validation warning: *Those passwords didn't match. Please try again*, and not completing the sign up process.

Signup successfully passes validations and creates user:
Tests that a user can successfully sign up for a new account by entering valid values into the fields and clicking the *'Create My Account'* button.

TrustedAIProviders.test.js

This suite tests that the Trusted AI Providers section of the results page successfully renders and displays a list of Trusted AI Providers specified by the Admin to the user **(US 4.03).**

Trusted AI Providers renders:
Tests that the Trusted AI Providers table successfully renders and displays the data for providers.

Trusted AI Providers renders with no data:
Tests that the Trusted AI Providers table successfully renders with no data

UserSettings.test.js

This suite tests the functionality of the user settings dropdown menu, which allows a user to navigate back to the home screen, admins to access the admin panel, users to change their account information (email, username, password), and update their organization.

User Settings renders successfully:
Tests that the user settings dropdown menu renders properly for logged in users.
User Settings transitions to Change Email successfully:
Tests that a user can edit the email address associated with their account by clicking on the USer Settings button and selecting the *'Change Email'* button.

User Settings submit Change Email fails validation:
Tests that a user cannot update their email address to an invalid email address and ensures that a validation warning: *Invalid Email* is displayed and does not update the email address.

User Settings submit Change Email passes validation:
Tests that a user can change their email address by entering a valid address, confirming their password, and clicking the *'Submit'* button.

User Settings transitions to Change Username successfully:
Tests that a user can change/add their username by pressing the *'Change Username'* button.

User Settings submit Change Username fails validation:
Tests that a user cannot update their username to an invalid one and ensures that a validation warning: *Invalid Username* is displayed and does not update the username.

User Settings submit Change Username passes validation:
Tests that a user can change their username by entering a valid one, confirming their password, and clicking the *'Submit'* button.

User Settings transitions to Change Password successfully:
Tests that a user can change their password by pressing the *'Change Password'* button.

User Settings submit Change Password fails validation:
Tests that a user cannot update their username to an invalid or weak password and ensures that a validation warning: *Weak Password* is displayed and does not update the password.

User Settings submit Change Password passes validation:
Tests that a user can change their username by entering a new valid password, confirming their previous password, and clicking the *'Submit'* button.

Selenium Test

Online testing suites to test the online running app. All automated Selenium tests are found in the `frontend/src/tests/integration_tests/` project directory.

AdminPage.test.js

This suite tests the various functionalities of the Admin page. The test is set up by logging in as to an account with Admin credentials.

admin can access admin panel:
Tests that a logged in admin user can access the administration panel through the dropdown user settings menu in the top right corner. Confirms that the private route is accessible to admin users.

admin can create a new questions ***(US 5.07)****:*
Tests that an admin user can manage the survey questions by creating a new one. Fires the add question button and confirms the new question modal pops up for editing.

admin can edit a question ***(US 5.07)****:*
Tests that an admin user can manage the survey questions by editing an existing one. Fires the edit button associated with a question row, and confirms the data loads into the pop for editing.

admin can create a new Trusted AI Provider ***(US 5.09)****:*
Tests that an admin user can add a new Trusted AI Provider source to the existing pool. Clicks on the *Providers* tab to navigate to the table, and clicks the add button to confirm a window pops up where they can add a new Provider.

admin can edit a Trusted AI Provider ***(US 5.09)****:*
Tests that an admin user can edit a Trusted AI Provider source to the existing pool. Clicks on the *Providers* tab to navigate to the table, and clicks the add button to confirm a window pops up where they can add a new Provider.

admin can view analytics ***(US 5.06)****:*
Tests that the admin can view basic site analytics. Clicks the *Analytics* tab to open the analytics panel and confirms the analytics information renders.

admin can view a list of users ***(US 5.04)****:*
Tests that the admin has access to a list of registered users, so they can manage and delete accounts as necessary, and view user submissions. Clicks the *Users* tab to open the users page and confirms a table of users renders

admin can view a list of submissions ***(US 5.08)****:*
Tests that the admin has access to the submissions of all non-registered users, so they have access to responses for review, without the user having to email them their specific responses. Clicks the *Submissions* tab which shows them a list of automatically saved survey submissions stored in the database that do not have an associated account.

admin can view a response ***(US 5.04, US 5.08)****:*
Tests that the admin has can view the specific responses of a survey submissions (either of a logged in account or not logged in user). It selects a specific submission by clicking on the *Submission* button and confirming it redirects to the *Results* page, where they can view the submissions report card.

HomePage.test.js

This suite tests the actions a user can take from the Home page of the App including registering, logging in and out, viewing existing surveys once logged in, and starting a survey without logging in.

user can open User Registration page ***(US 3.01)****:*
Tests that a user can register for an account by clicking the *Login* button, clicking the *Create your account* link, and transitioning to the registration page.

user can create an account ***(US 3.01)****:*
Tests that a user can register for an account by filling out the registration form with valid data and clicking *Create My Account*.

user can successfully open and Log into selenium test user ***(US 3.02)****:* Tests that a registered user can log into their account by clicking the *Login* button, entering their account details (username and password) and clicking *Login*. They will then have access to the user settings dropdown menu.

user can see list of existing surveys once logged in ***(US 3.02)****:*
Tests that a logged in user is presented with a list of their previously started or completed surveys.

user can log out of their account ***(US 3.02)****:*
Tests that a logged in user can log out of their account through the user dropdown menu.

user can start a new survey without logging in ***(US 1.05)****:*
Tests that a user who is not logged in can still access and completed the survey by simply clicking the *Start New Survey* button.

ResultsPage.test.js

This suite tests the functions of the Results page of the App including viewing Scores, Report Card, a list of Trusted AI Providers and Trusted AI Resource links.

User can switch between Tabs ***(US 4.03)****:*
Tests that a user can navigate to the Results page and view the different tabs to view the survey score, report card, and lists of Trusted AI Providers and Resources by clicking on the different tabs on the page.

User can export the survey to a csv file:
Tests that a user can export the report card for their current submission as a csv by clicking the *Export CSV* button.

User can export the survey to a pdf file ***(US 4.04)****:*
Tests that a user can export the report card for their current submission as a pds by clicking the *Export* button.
User can start again from results page:
Tests that a user can start a new survey from the Results page by clicking on the *Start Again* button, which will redirect them to the home page.

SurveyPage.test.js

This suite tests the different actions a user can take during the completion of a survey, including navigating to different dimensions from the navigation menu, filtering the survey by selecting filters for Role, Industry, Region, and Lifecycle, save their submission progress, and finish their survey to have it saved in the database, and redirect them to the results page.

*User can navigate to different dimensions and questions **(US 1.01, US 1.02)**:* Tests that a user has access to a navigation menu that they can click on to navigate to the different dimensions/questions in the survey. By clicking the *Accountability* menu, they are presented with indicators for each question in the Accountability dimensions (which are coloured grey in unanswered, and blue if answered). By clicking on the question indicators, the survey jumps the user to this page.

*User can filter by their role, industry, region, lifecycle **(US 2.01, 2.02, 2.03)**:* Tests that a user can filter the survey questions based on their role, region, industry and project lifecycle. By clicking the *filters* menu, and a dropdown list is present to them where they can select multiple filters for each category. The filters are then applied by clicking the *Apply Filters* button.

User can save their results:
Tests that a logged in user can save their progress at any point during the survey by clicking the *Save* button.

User can finish the survey **(US 5.08)***:*
Tests that a user can finish the survey at any point by clicking the *Finish* button. If the user is logged in, the submission is saved to the account submission database so the admin can view their submissions from the admin panel. If the user is not logged in, the submission is automatically saved into the no-user submission category so the admin can still view the results.

Not Tested or Developed

US 4.01 - Improve report card recommendations
Would Like - not implemented

US 4.02 - Suggested resources column in report card
Would Like - not implemented

US 5.10 - Add questions from spreadsheet
Would Like - not implemented