



## TDD: [Learner Diagnostic Tests](#)

Author: Nikhil Agarwal

Date: 01/04/2022 (created)

## Section 1: About You

What project are you applying for?

**Title:** Learner Diagnostic Tests.

Why are you interested in working with Oppia, and on your chosen project?

Working with Oppia is one of the most exciting tasks because its mission to provide and create lessons in a fun and exciting way fascinates me very much. After working with various projects regarding APIs, JSON handling, schemas, and request responses, this project seems interesting and committable.



Timezone: Indian Standard Time(IST) or GMT + 5:30

Time commitment

I plan to work approx 30 hours a week.

Essential Prerequisites

Answer the following questions (for Oppia web GSoC students):

- I am able to run a single backend test target on my machine.

```
pre-push hook file is now executable!
-----
Tasks still running:
  core.domain.rights_manager_test.ExplorationRightsTests (started 23:41:41)
-----
18:12:27 FINISHED core.domain.rights_manager_test.ExplorationRightsTests: 45.4 secs

+-----+
| SUMMARY OF TESTS |
+-----+

SUCCESS  core.domain.rights_manager_test.ExplorationRightsTests: 32 tests (38.1 secs)

Ran 32 tests in 1 test class.
All tests passed.

Done!
nikhil@nikhil-HP-Laptop-15-ds0xxx: ~/oppia/oppia$
```

- I am able to run all the frontend tests at once on my machine. (Show a screenshot of a successful test.)

```
LOG: 'Spec: Question misconception editor component should enable edit mode correctly has passed'
Chrome Headless 88.0.4324.182 (Linux x86_64): Executed 4340 of 4347 SUCCESS (0 secs / 1 min 16.925 secs)
ERROR: 'Error communicating with server. Please try again.'
Chrome Headless 88.0.4324.182 (Linux x86_64): Executed 4341 of 4347 SUCCESS (0 secs / 1 min 16.933 secs)
LOG: 'Spec: Question misconception editor component should tag a misconception correctly has passed'
Chrome Headless 88.0.4324.182 (Linux x86_64): Executed 4341 of 4347 SUCCESS (0 secs / 1 min 16.933 secs)
LOG: 'Spec: Question misconception editor component should use feedback by default has passed'
Chrome Headless 88.0.4324.182 (Linux x86_64): Executed 4342 of 4347 SUCCESS (0 secs / 1 min 16.942 secs)
LOG: 'Spec: Question misconception editor component should update tagged misconception name correctly has passed'
Chrome Headless 88.0.4324.182 (Linux x86_64): Executed 4343 of 4347 SUCCESS (0 secs / 1 min 16.951 secs)
LOG: 'Spec: Question misconception editor component should initialize correctly when tagged misconception is provided has passed'
Chrome Headless 88.0.4324.182 (Linux x86_64): Executed 4344 of 4347 SUCCESS (0 secs / 1 min 16.961 secs)
ERROR: 'Error communicating with server. Please try again.'
Chrome Headless 88.0.4324.182 (Linux x86_64): Executed 4345 of 4347 SUCCESS (0 secs / 1 min 16.971 secs)
LOG: 'Spec: Question misconception editor component should not tag a misconception if the modal was dismissed has passed'
Chrome Headless 88.0.4324.182 (Linux x86_64): Executed 4345 of 4347 SUCCESS (0 secs / 1 min 16.971 secs)
LOG: 'Spec: Question misconception editor component should report containing misconceptions correctly has passed'
Chrome Headless 88.0.4324.182 (Linux x86_64): Executed 4346 of 4347 SUCCESS (0 secs / 1 min 16.982 secs)
Chrome Headless 88.0.4324.182 (Linux x86_64): Executed 4347 of 4347 SUCCESS (1 min 27.923 secs / 1 min 16.991 secs)
TOTAL: 4347 SUCCESS
TOTAL: 4347 SUCCESS
08 04 2021 01:34:26.418:WARN [launcher]: ChromeHeadless was not killed in 2000 ms, sending SIGKILL.
Done!
-----
Frontend Coverage Checks Not Passed.
-----
...the component to seem to be not completely tested - make sure it's fully covered
```

- I am able to run one suite of e2e tests on my machine. (Show a screenshot of a successful test.)

```
PageNotFoundException

.    ? should be correct for voice artists

5 specs, 0 failures
Finished in 550.517 seconds

Executed 5 of 5 specs SUCCESS in 9 mins 11 secs.
[09:48:37] I/launcher - 0 instance(s) of WebDriver still running
[09:48:37] I/launcher - chrome #01 passed

t emulators: Received SIGTERM for the first time. Starting a clean shutdown.
```

Other summer obligations

None

Communication channels

Weekly meetings with the mentor for discussing the project progress and doubts.

## Overview

This TDD explains the implementation of the Learner Diagnostic Tests that will allow the learner to test their knowledge and get the topics recommendations.

## Problem Statement

Each learner that comes onto Oppia's platform has a different level of experience and comprehension within a given subject and some of them have trouble identifying which topic they should start within the Oppia Classroom as a result, Learner Diagnostic Tests will allow the learner to test their knowledge and get a set of recommendations from where they should begin learning.

<b>Link to PRD</b>	<a href="#">Learner Diagnostic test PRD</a>
<b>Target Audience</b>	Learners, generally between 7 and 14 years old. (So the target audience contains a wide set of learners with different levels of experience and comprehension, some may contain previous knowledge and some may not.)
<b>Core User Need</b>	As a learner, I need a way to know which topic (or topics) I should start with on the Oppia platform based on what I already know and understand about the various topics available in a classroom.
<b>What goals do we want the solution to achieve?</b>	<ol style="list-style-type: none"><li>1. Increased learner engagement - especially for first-time visitors of Oppia.</li><li>2. Increased lesson completion rates (i.e. lessons completed / lesson attempts).</li><li>3. Increased learner satisfaction - especially for first-time visitors of Oppia.</li></ol>

## Section: What

### Key user stories

#	Title	User Story Description (role, goal, motivation) "As a ..., I need ..., so that ..."	Priority <sup>1</sup>	List of tasks needed to achieve the goal (this is the "User Journey")	Links to mocks/prototypes, and/or PRD sections that spec out additional requirements.
1	Learner Diagnostic Tests	As a learner, I need guidance on which lessons in Math Classroom to start with so that I can begin learning on Oppia with something that's at the right level for me.	Must have	<p>The Learner sees a button to take the diagnostic test for the Oppia Math Classroom.</p> <p>The learner plays through the diagnostic test and answers the questions without hints, solutions, or feedback. Also, the first answer provided by the learner is marked for evaluation.</p> <p>On completing the test, the learner receives 0 (i.e. learner understands everything taught in the topics contained within the Classroom), 1, or 2 recommendations for which topic(s) to start with based on the accuracy of their answers.</p>	<a href="#">Mocks related to learners.</a>

<sup>1</sup> Use the **MoSCow** system ("Must have", "Should have", "Could have"). You can read more [here](#).

2	Diagnostic test discoverability	As a learner, I need to see that there is a "learner diagnostic test" feature so that I can be aware that it's available and take the test if it's useful for me.	Should have	<p>The first time a user views the classroom page with this new functionality, a small popover/modal will present, letting them know that they can use the new feature to get recommendations for which lesson to get started with.</p> <p>Learners should be able to dismiss this popover/modal manually. If they don't, it should disappear automatically after 2 minutes.</p>	
3	Topic editor restrictions for diagnostic test programming.	As a topic editor, I need to set up the skills in my topic so that I can ensure that the diagnostic tests adequately verify topic mastery	Should have.	<p>The topic editor, view a new card with the name "Diagnostic test" on the topic editor page. The card will display all the skill descriptions (only skills which are associated with subtopics i.e., uncategorized skills are not included) for the skills which are present in the corresponding topic.</p> <p>The topic editor can select upto 3 skills for the diagnostic test.</p>	<a href="#">Mocks related to the topic editor.</a>
4	Curriculum admin task for diagnostic test programming.	As a curriculum admin, I need to specify the dependency between the topics, so that I can ensure that learners test their skills gradually.	Must have	<p>The curriculum admins must have a new classroom admin page from which they can manage the task related to a classroom.</p> <p>The curriculum admins can edit the configuration related to a classroom and also provide dependencies between the topics for creating the diagnostic test.</p>	<a href="#">Mocks related to curriculum admins.</a>

## Technical requirements

### Additions/Changes to Web Server Endpoint Contracts

#	Endpoint URL	Request type	New / Existing	Description of the request/response contract
1	/diagnostic_test/<classroom_url_fragment>/	GET	New	The Main HTML page for the diagnostic test is requested using this URL.
2	/diagnostic_test_result	GET	New	The HTML page for the diagnostic test result page.
3	<a href="#">/topic_editor_handler/data</a>	PUT	Existing	A new field, 'diagnostic_test_skill_ids' should be passed through 'topic_and_subtopic_page_change_dicts' while saving 'the topic.
4	/diagnostic_test_topics_handler/<classroom_url_fragment>/	GET	New	Fetches the diagnostic test data (a dict with topic_name as key and parent topic name as value) for a particular classroom.
		PUT	New	Curriculum admins should be able to create the dependency tree between the topics and save the DAG.
5	/classroom-admin	GET	New	The curriculum admin should be able to get the main HTML page of classroom admin, from where he can manage all the tasks related to the classroom.
6	classroom_config_property_handler/<classroom_url_fragment>	PUT	New	The curriculum admin should be able to manage config properties related to a classroom.
		GET		



## Call to webserver endpoints

#	User	Endpoint URL	Request type	Description
1.	Learner	/diagnostic_test	GET	The Main HTML page for the diagnostic test is requested using this URL.
		/diagnostic_test_result	GET	The HTML page for the diagnostic test result page.
		/diagnostic_test_data/<classroom_url_fragment>/<topic_url_fragment>	GET	Some set of questions is presented to the learners, in the diagnostic test page, using this URL.
		<a href="#">/question_player_handler</a>	GET	Used to fetch the questions based on skill_ids, for the diagnostic test.
		<a href="#">/learn/&lt;classroom_url_fragment&gt;/&lt;topic_url_fragment&gt;</a>	GET	Learners click the recommended topic (after giving the diagnostic test) and navigated to the topic player page.
2.	Topic editor	<a href="#">/topic_editor_handler/data</a>	PUT	<p>A new field, diagnostic_test_skill_ids should be passed through 'topic_and_subtopic_page_change_dicts' while saving the topic.</p> <p>(The change dict will reflect changes on Topic class, hence one more field diagnostic_test_skill_ids, should be added in the domain and model layer. I will cover this in more detail in the "How" section of the design doc.)</p>

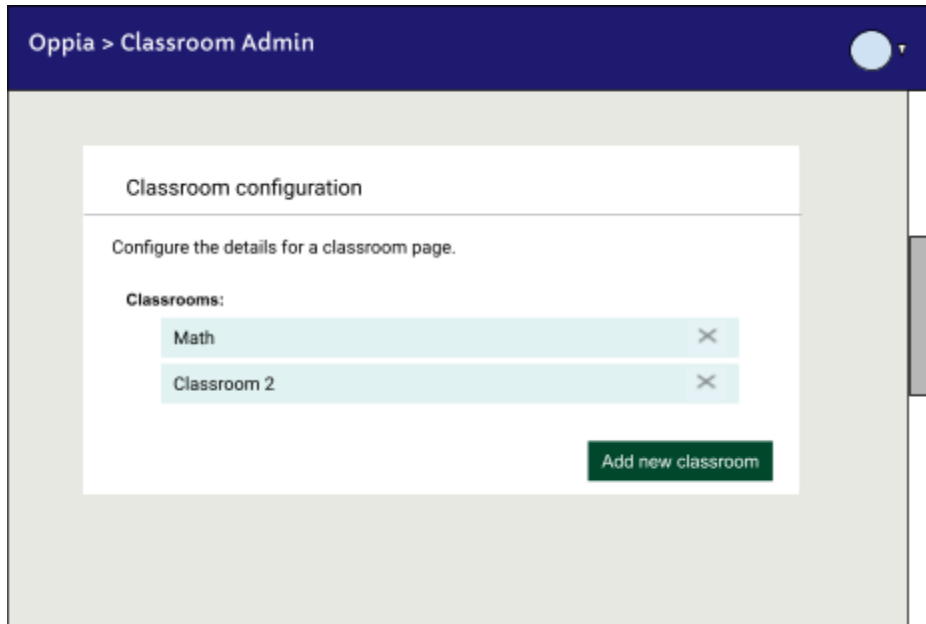
		<a href="#">/rightshandler/change_topic_status/&lt;topic_id&gt;</a>	PUT	The topic editor publishes the topic, using this API request.
3	Curriculum admin	/classroom-admin	GET	The curriculum admin should be able to get the main HTML page of classroom admin, from where they can manage all the tasks related to the classroom.
		/daignostic_test_topics_handler/<classroom_url_fragment>/	GET	Fetches the diagnostic test data (a dict with topic_name as key and parent topic name as value) for a particular classroom.
			PUT	Curriculum admins should be able to create the dependency tree between the topics and saves the tree.
		/classroom_config_property_handler/<classroom_url_fragment	PUT & GET	The curriculum admin should be able to manage config properties related to the classroom.

UI screen components

Mock related to the Curriculum admin

#	Mock	Transition
---	------	------------

1



The classroom admin page contains a card that contains the tiles for all the different classrooms in Oppia.

Curriculum admins can create another classroom by clicking the button "Add new classroom".

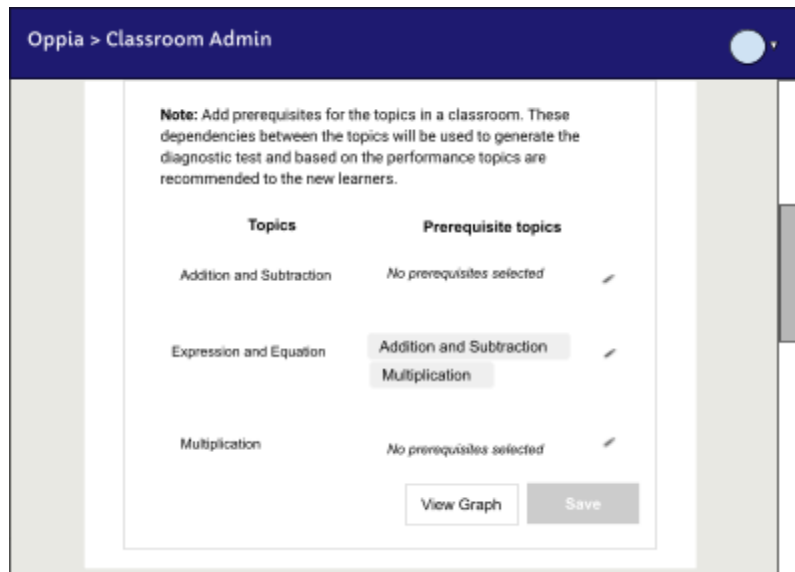
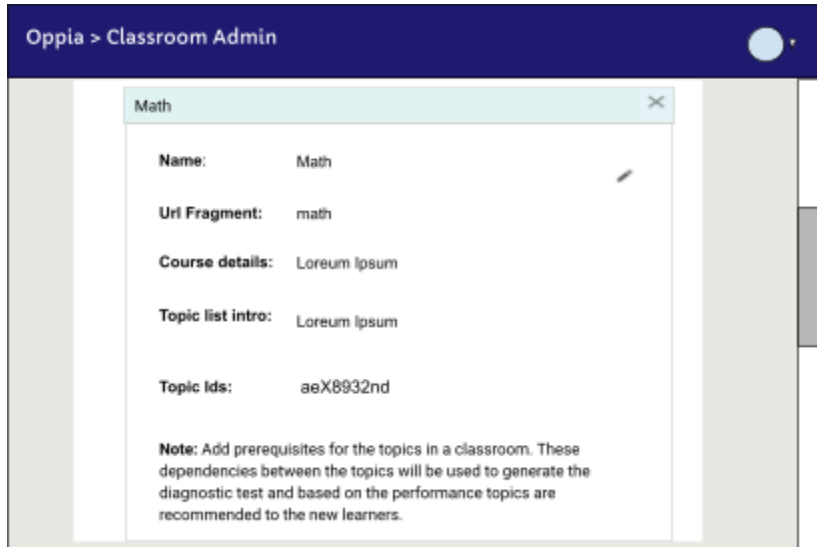
(The case to add a new classroom will be handled in the later section (mock 3))

They can also remove the existing classroom configurations by clicking the "X" icon on the right of each tile.

Clicking the "X" button, a modal will be popped up asking for the confirmation i.e., whether they intended to delete the tile or they pressed the "X" button unintentionally. (The confirmation modal will be consistent with the other pages of Oppia.)

Clicking any of the tiles will open more views of a classroom. (The mocks after clicking the Math tile are given below.)

2



After clicking a tile a more detailed view of a classroom opens. The view primarily contains two sections:

- The first section contains basic classroom details like name, url\_fragment, course\_details, etc.
- The second section contains a list of topics and their prerequisites that were present in a classroom.

**Note:** One point to ignore is that the first section contains 1 topic id (aeX8932nd), while in the second section, I have presented the 3 topics (Addition and Subtraction, Expression and Equation, Multiplication).

In the second section, I only mean to represent several topics in a classroom, while ignoring some of the detailing.

3

The screenshot shows a web interface for 'Oppia > Classroom Admin'. A modal window titled 'Math' is open, containing a form with the following fields and values:

- Name:** Math
- Url Fragment:** math
- Course details:** Lorem Ipsum
- Topic list intro:** Lorem Ipsum
- Topic Ids:** aeX8932nd (with a clear 'X' button)

At the bottom of the form are three buttons: 'Add Topic' (disabled), 'Cancel', and 'Save'.

After clicking the pencil icon for the first section, this view will be presented.

The name section is two-way data binding, i.e., the name written on the form will simultaneously be presented in the top classroom tile (sky blue color section).

This will be important when the user creates the new classroom, all the sections will be empty and as they fill up the name the header tile name will be updated in real-time.

4

The screenshot shows the 'Oppia > Classroom Admin' interface. At the top, there is a dark blue header with the text 'Oppia > Classroom Admin' and a small blue circle icon. Below the header, a note reads: 'Note: Add prerequisites for the topics in a classroom. These dependencies between the topics will be used to generate the diagnostic test and based on the performance topics are recommended to the new learners.' The main content area is a table with two columns: 'Topics' and 'Prerequisite topics'. The table has three rows. The first row is for 'Addition and Subtraction' with 'No prerequisites selected' and a pencil icon. The second row is for 'Expression and Equation' with 'Addition and Subtraction' and 'Multiplication' listed as prerequisites and a pencil icon. The third row is for 'Multiplication' with 'No prerequisites selected' and a pencil icon. At the bottom of the table, there are two buttons: 'View Graph' and 'Save'.

Topics	Prerequisite topics
Addition and Subtraction	No prerequisites selected
Expression and Equation	Addition and Subtraction Multiplication
Multiplication	No prerequisites selected

The second section contains two columns one for the topics in the classroom and the other for the prerequisites of the corresponding topics.

Hovering on any row on the right column, will change the shade of the zone which contains the prerequisites of the topics.




Clicking on the pencil icon or anywhere in the shaded zone will enable editing for the corresponding topic/row.

The save button is disabled initially, because there is nothing to save.

5

Oppia > Classroom Admin

**Note:** Add prerequisites for the topics in a classroom. These dependencies between the topics will be used to generate the diagnostic test and based on the performance topics are recommended to the new learners.

Topics	Prerequisite topics
Addition and Subtraction	No prerequisites selected 
Expression and Equation	<div style="background-color: #e0e0e0; padding: 2px;">Addition and Subtraction</div> <div style="background-color: #e0e0e0; padding: 2px;">Multiplication</div> 
Multiplication	<div style="border: 1px solid #ccc; padding: 5px;">           Select Prerequisites  <ul style="list-style-type: none"> <li><input checked="" type="checkbox"/> Addition and Subtraction</li> <li><input type="checkbox"/> Expression and Equation</li> </ul> </div>

## Steps:

1. On clicking the shaded zone for any topic, an inline view to select prerequisites will be opened.
2. From this, the curriculum admins can select or deselected prerequisites.
3. After selection, they can click the 'X' icon to exit from the editing mode.

## Notes:

1. While the curriculum admins are editing a topic they will not be able to see the graph because until they don't finish editing for a row the changes, the change will not be reflected on the graph.
2. Only one row is editable at a time, i.e., if any of the inline selector for the prerequisites is opened, then other topics will not be edited.
3. In case they selected prerequisites in such a way that the prerequisites are interdependent on each other, in that case, an error message will be presented.
  - a. Example: "There is a cycle in the prerequisites: **Topic 1** depends on **Topic 2** which depends on **Topic 1**."
4. Also Present an error message for reducing the edges of the graph, for the case when "A depends on B, B depends on C, therefore A depends on C".
  - a. Example: Remove redundant connections between Topic 1

and Topic 3. Since Topic 3 is dependent on Topic 2 and Topic 2 is dependent on Topic 1, so no need to connect Topic 3 and Topic 1.

6

The screenshot shows the 'Oppia > Classroom Admin' interface. At the top, there is a dark blue header with the text 'Oppia > Classroom Admin' and a circular profile icon. Below the header, a note reads: 'Note: Add prerequisites for the topics in a classroom. These dependencies between the topics will be used to generate the diagnostic test and based on the performance topics are recommended to the new learners.' The main content area is a table with two columns: 'Topics' and 'Prerequisite topics'. The table lists three topics: 'Addition and Subtraction', 'Expression and Equation', and 'Multiplication'. For 'Addition and Subtraction', no prerequisites are selected. For 'Expression and Equation', 'Addition and Subtraction' and 'Multiplication' are listed as prerequisites. For 'Multiplication', 'Addition and Subtraction' is listed as a prerequisite. Each row has a pencil icon to its right, indicating editability. At the bottom of the table, there are two buttons: 'View Graph' and 'Save'.

Topics	Prerequisite topics
Addition and Subtraction	No prerequisites selected
Expression and Equation	Addition and Subtraction Multiplication
Multiplication	Addition and Subtraction

View Graph Save

After adding the prerequisites for the "Multiplication", the save button will be enabled. This allows the curriculum admins to save the changes.



7



The figure contains a sample graph.

After providing the prerequisites for topics, the curriculum admins can view the graph, by clicking the button “View Graph”.

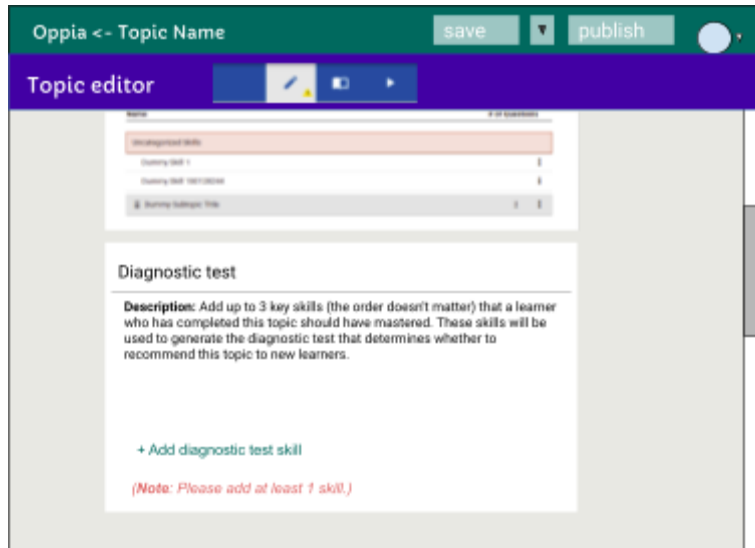
Clicking the button “View graph”, will result in opening a new modal in the same tab. (Similar to exploration states-graph)

After viewing the graph the curriculum admins can close the modal by clicking the close button.

Mock related to the Topic editor

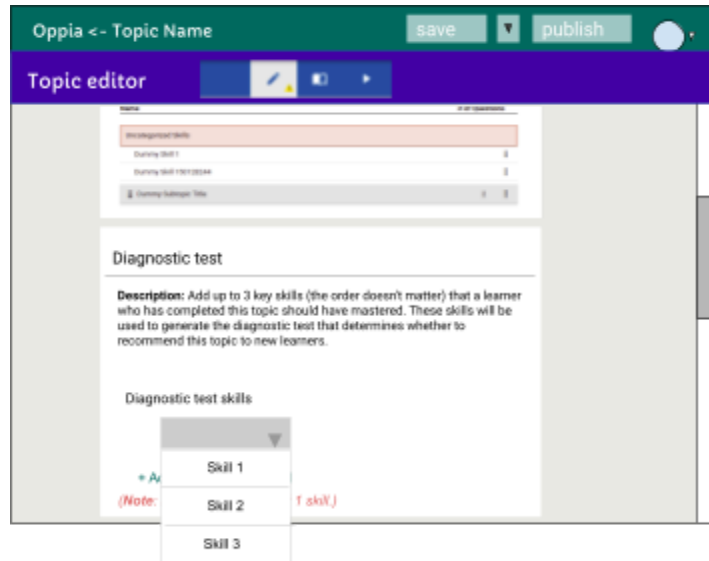
#	Mock	Transition
---	------	------------

1



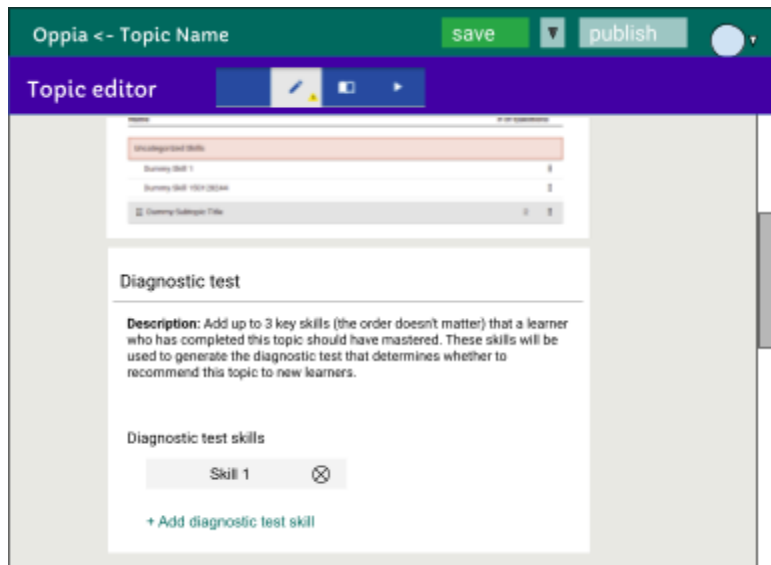
1. The topic editors see a new card with the name "Diagnostic test" on the topic editor page. They can select the skills for the diagnostic test using this card.
2. The diagnostic test card contains a description: "Add up to 3 key skills (the order doesn't matter) that a learner who has completed this topic should have mastered. These skills will be used to generate the diagnostic test that determines whether to recommend this topic to new learners."
3. A note will be presented to the learner which guides them to add at least one skill for the diagnostic test. A note stating: "Please add at least 1 skill."
4. Disable the publish button and a warning message will be added when no skills are added for the diagnostic test. Presenting the warning, will be similar to the current implementation i.e., adding a message in the issue and presenting the issue count in a small yellow triangle below the pencil icon on the topic editor navbar, and on hovering the yellow icon the issue list will be presented.

2




1. Clicking the “Add diagnostic test skill”, button, presents a dropdown to select skills for the diagnostic test.

3



1. After selecting a skill, a tile containing the skill description will be presented, and the topic editors can also remove the tile by clicking the cross icon (X) in the tile.
2. Note: After selecting three questions, the option to add a skill for the diagnostic test will disappear.

Mock related to the Learners

#	Mocks	Notes
1	 <p>The screenshot shows the PPIA website interface. At the top, there is a navigation bar with links for HOME, LEARN, ABOUT, GET INVOLVED, and DONATE, along with a language selector set to ENGLISH. The main content area is titled 'Topics Covered' and features two primary options: 'New to maths?' which suggests starting from the basics with 'Place Value', and 'Already know some topics?' which offers a 5-minute test. Below these are three topic cards: 'Place Values' (5 Lessons) with a number line diagram, 'Addition and Subtraction' (7 Lessons) with a bead bar diagram, and 'Multiplication' (7 Lessons) with a multiplication diagram showing <math>5 \times 3 = 15</math>. A search bar and category/language filters are also visible.</p>	<p>On the math classroom page two new clickable items were added.</p> <ol style="list-style-type: none"><li>New to maths?<ul style="list-style-type: none"><li>“Start from the basic from our first topic, Place Value.”</li></ul></li><li>Already know some topics?<ul style="list-style-type: none"><li>“Check your level, by giving a test. The test is about 5 mins to complete.”</li></ul></li></ol> <p>These two sections consider two different sets of learners: The one who is very new to these topics, and the one who has some skills but they are not sure where to start with.</p> <p><b>Note:</b> The learners who have some skills and also know which topic they want to learn are not addressed because they are expected to jump directly to the topic and start learning.</p> <p>Clicking the first item “New to maths?” will start the place value topic because they are expected to start with the first topic if they are very new.</p> <p>Clicking the second item “Already know some topics?” will take a test to check their understanding of the existing topics and based on the performance 1 or 2 topics were recommended.</p>

2

Oppia

HOME LEARN ABOUT GET INVOLVED DONATE ENGLISH

Topics Covered

Get topic recommendations based on your current knowledge! ✕

**New to maths?**  
Start from the basic from our first topic, Place Value.

**Already know some topics?**  
Check your level, by giving a test. The test takes about 5 mins to complete.

**Place Values**  
5 Lessons

**Addition and Subtraction**  
7 Lessons

**Multiplication**  
7 Lessons

Explore More Lessons Made by the Community  
Search through our Community Library

What are you curious about? All Categories All Languages

ABOUT OPPIA  
About  
The Oppia Foundation  
Blog  
Forum

TEACH US  
Get Started  
Teacher Resources  
Self-Paced Tutorials  
Browse the Library  
Helpdesk

CONTRIBUTE  
Volunteer  
Donate  
Participate in our  
Private Policy

FOLLOW US  
YouTube  
Facebook  
Twitter

A popover will be presented to the learner if they are coming to the math classroom page for the first time.

The popover will automatically disappear after two mins.

3

Oppia > Learner Diagnostic Test

HOME LEARN ABOUT GET INVOLVED DONATE ENGLISH

## Learner Diagnostic Test

25% Complete

What fraction is shown by the shaded part of this drawing?

Enter a fraction in the form  $\frac{x}{y}$

I DON'T KNOW SUBMIT

Question presentation view.

Available Options:

- The learner clicks the “Submit” button to answer the current question and continue the test.
- The learner clicks the “I don’t know” option to skip the current question.
- Leaving the test will present a modal for confirmation because the progress will not be saved.
  - The modal will be similar to other pages in Oppia.

4

The screenshot shows the Oppia Learner Diagnostic Test interface. At the top, there is a navigation bar with the Oppia logo and the text "Learner Diagnostic Test". To the right of the logo are links for "HOME", "LEARN", "ABOUT", "GET INVOLVED", and "DONATE". Further right is a language selector set to "ENGLISH" and a gold coin icon. Below the navigation bar is a blue header with the text "Learner Diagnostic Test" and a repeating pattern of icons. A progress bar is shown with the text "25% Complete". The main content area features a question card with a cartoon character icon on the left. The question text is: "Great! said Mr. Baker. "Now, here's another question for you. If I write the fraction  $\frac{5}{9}$ , what does the '9' mean?" Below the question are three radio button options: "The number of chosen parts", "The number of equal parts that the whole cake is divided into", and "I can't remember!". At the bottom right of the question card is a green button labeled "I DON'T KNOW".

In the case of the MCQ questions, the submit button will not be provided to the learner.


5

Oppia > Learner Diagnostic Test

HOME LEARN ABOUT GET INVOLVED DONATE ENGLISH

**Test complete. Well done!**

Based on your answers in the diagnostic test, we recommend you start learning with this topic first.



Multiplication  
7 Lessons

Go to Classroom

This result page is presented after completion of the diagnostic test, this page contains 1 topic as a recommendation.



6

Oppia > Learner Diagnostic Test

HOME LEARN ABOUT GET INVOLVED DONATE ENGLISH

**Test complete. Well done!**

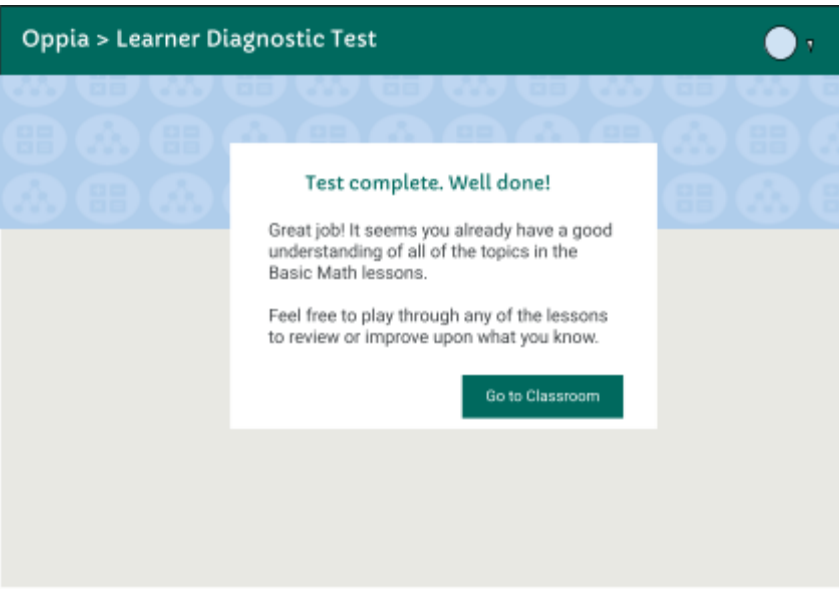
Based on your answers in the diagnostic test, we recommend you start learning with one of these topics first.

**Place Values**  
7 Lessons

**Multiplication**  
7 Lessons

Go to Classroom

This result page is presented after completion of the diagnostic test, this page contains 2 topics as a recommendation.

7		<p>This result page is presented after completion of the diagnostic test, and this page contains 0 topics as a recommendation. This implies that the learner knows everything related to the topics of a classroom.</p> <p>The text on the card contains: “Great job! It seems you already have a good understanding of all of the topics in the Basic Math lessons. Feel free to play through any of the lessons to review or improve upon what you know.”</p>
---	--	---

Table

#	ID	Description of new UI component	i18n required?	Mock/spec links	A11y requirements
1.	Learner diagnostic test	Diagnostic page presented to the learners	Yes	<a href="#">Mocks related to learners.</a>	Yes
2.	Topic editor page	Topic editor selects the skills for the diagnostic test from the topic editor page.	No	<a href="#">Mocks related to topic editors</a>	No

3	Classroom admin page	Curriculum admin should be able to create the diagnostic test from the classroom admin page.	No	<a href="#">Mocks related to curriculum admins</a>	No
---	----------------------	--	----	--	----

### Data Handling privacy

Data related to users is not stored as a part of this project.

#	Type of data	Description	Why do we need to store this data?	Anonymized?	Can the user opt-out?	Wipeout policy	Takeout policy
1	Classroom_url_fragment, topic_adjacency_list	Classroom_url_fragment: StringProperty  Topic_adjacency_list: JsonProperty with topic_id as key and list of prerequisites as a value.	This model stores hierarchical information on topics in the corresponding classroom.  (This model will be used to present questions in the diagnostic test.)				

### Existing Status Quo

Currently, the learner can go to the Math Classroom page and see the range of topics that are available to them, and then click on each topic to view the lessons within that topic. However, there's no way to know exactly which skills are taught in each topic, so a learner must click into each lesson and play through one or more lesson cards to best determine whether the lesson is a good fit for them.

Pros: Learners can determine for themselves which lessons to start with and get a sense of the range of lessons that are available to them.

Cons: Takes a lot of time and persistence on the part of the learner.

## Solution Overview

A feature by which learners can give the diagnostic test to get zero or one or two topic recommendations, from which they can start their learning journey with Oppia.

The skills from each topic are selected by the topic editor and the dependency between the topics for a classroom is provided by the curriculum admins. Hence the diagnostic test functionality, primarily involves **topic editors, curriculum admins & learners**.

## Third-Party Libraries

No third-party libraries are used in this project, a majority of the methods and services (frontend and backend) already exist in the codebase.

## “Service” Dependencies


No external services are used in this project.


## Impact on Other Oppia Teams

The learner feedback team will benefit from this feature since students will be able to figure out their recommended topic in a self-service way.

## Key High-level and Architectural decisions

Architecture to store the diagnostic test data

: Creation of a new model.

: Modification in the existing model.

### DiagnosticTestModel (Model 1)

Fields	Description	Intention	Priority
Classroom_url_fragment, topics_adjacency_list	Classroom_url_fragment: StringProperty  Topic_adjacency_list: JsonProperty with topic_id as key and list of prerequisites as a value.	This model stores the topics DAG for a classroom.	<b>Must have</b>

**Note:** Below two models are alternatives to each other, so their pros and cons b/w them are discussed after these two tables.

### TopicModel (Model 2)

Fields	Description	Intention	Priority
skill_ids_for_diagn ostic_test	Skill_ids_for_diagnostic_test: JsonProperty (A list of skill_ids for the diagnostic test.)	These skills are used for creating diagnostic test questions for the corresponding topic.	<b>Should have.</b>

### TopicSkillsLinkForDiagnosticTestModel (Model 3)

Fields	Description	Intention	Priority
id,	Id: topic_id,	A model for linking topic_id with the list of	<b>Should have.</b>

skill_ids_for_diagnostic_test	Skill_ids_for_diagnostic_test: JsonProperty (A list of skill_ids for the diagnostic test.)	skill_ids, which are used by the diagnostic test.	
-------------------------------	--	---	--

Pros and Cons b/w model 2 and 3

User	Need	Steps performed in Model 2 approach	Steps performed in Model 3 approach
<b>Topic editor</b>	Should be able to view all the skill descriptions present in the subtopics.  (Skills that are present in the topic.subtopics)	Collection of all the skill_ids from the corresponding topic. <ul style="list-style-type: none"> <li>No backend call (since the list of skills will be stored in the topic itself.)</li> </ul>	Collects the list of skill_ids from "TopicSkillsLinkForDiagnosticTestModel" using get_by_id() call. <ul style="list-style-type: none"> <li>1 GET call, to receive the list of skill_ids for the corresponding topic.</li> </ul>
		Fetch skill description from the list of skills, collected from the above step. <ul style="list-style-type: none"> <li>1 get_multi() call to SkillSummaryModel.</li> </ul>	
		The topic editor selects some of the skills (max 3) after reading the description for each skill and publishes the topic. <ul style="list-style-type: none"> <li>1 put() call for saving the topic in the datastore.</li> </ul>	The topic editor selects some of the skills (max 3) after reading the description for each skill and publishes the topic. While publishing the selected skills are saved to "TopicSkillsLinkForDiagnosticTestModel". <ul style="list-style-type: none"> <li>1 put for saving topic skill link.</li> </ul>
<b>Conclusion</b>		1 get_multi, 1 put	1 get, 1 get_multi, 1 put

<b>Curriculum admins</b>	Curriculum admins will not use TopicModel or TopicSkillsLinkForDiagnosticTestModel for their task, they will use DiagnosticTestModel solely.		
<b>Learners</b>		Learners will use TopicModels to get the list of skills for the diagnostic test and based on the skill_ids, the questions will be fetched.	Learners will use TopicSkillLinkForDiagnosticTestModel to get the list of skills for the diagnostic test and based on the skill_ids, the questions will be fetched.
<b>Conclusion</b>		.In both the model structures the number of GET calls will be equal for the learners.	

**Result:** Comparing the datastore and backend API calls with respect to different users, the model 2 approach i.e., storing the diagnostic\_test\_skill\_ids field into the topic model is a better approach because of the following reasons:

1. Lesser # backend API calls.
2. Lesser # get and get\_multi calls to the datastore.
3. Currently, the topic models are versioned, so the model 3 approach will also require an extra field "version" for maintaining data consistency. Thus maintaining a version in a single model (TopicModel) is easier than maintaining versions in two different models (TopicModel and TopicSkillsLinkForDiagnosticTestModel).

#### Reason for selecting 3 questions from a topic

1. For current scope: The maximum number of topics in a classroom is 15
  - a. This number indicates that for testing a wider range of topics the number of questions from each topic should be less.
2. Based on the PRD, each topic contains a maximum of 3 skills for the diagnostic test. Thus testing at least 1 question from each skill is the marker to guess whether the learner is holding the knowledge of that skill or not.
  - a. In some cases, 1 question for testing a skill may not be a good fit to decide whether the learner holds the knowledge for the skill or not. But considering the maximum number of questions (~ 15) and future plans (i.e., creating a diagnostic test on a topic for recommending a story), 1 question is sufficient for getting the initial recommendations for the new learners.
3. Also, for a topic, even if the learner attempts any question wrongly they get an equal opportunity to prove their knowledge in the other two questions. Based on the performance, a decision is made whether the topic should be recommended or not.

## Question selection strategy

### Frontend-Backend Communication for questions fetching

1. [Frontend]: Initialization of the player page leads to fetching of the question from the question-backend-api.service
  - a. The [fetchQuestion](#) method takes 3 params
    - Skill\_ids: List of skill\_ids that will be used to present questions in the diagnostic test.
    - Question\_count: The number of questions that will be presented from these skills i.e., # questions to test a topic.
    - Fetch\_by\_difficulty: The difficulty level of the questions. The **True** value indicates that the questions will be of medium level.
2. [Backend]: [QuestionPlayerHandler](#) in the backend will accept the frontend call and provide the questions as per skill\_ids.
  - a. Service: [get\\_questions\\_by\\_skill\\_ids\(\)](#)
  - b. Model: [get\\_question\\_skill\\_links\\_based\\_on\\_difficulty\\_equidistributed\\_by\\_skill\(\)](#).

**Note:** For diagnostic-test-player, the fetch\_by\_difficulty field should be true for fetching medium-level questions and the get\_question\_skill\_links\_based\_on\_difficulty\_equidistributed\_by\_skill() method will be used for equi distributing the questions to the number of skills.

### Question selection criteria

#### **Terminology:**

1. Intrinsic option – The option which is not available for the user but they are present and triggered automatically based on some conditions.

#### **Special consideration:**

1. If a learner fails in a question, then another question from the same skill will be presented to the learner. This intrinsic option is like a “**lifeline**” because each diagnostic test skill from a topic should be passed in order to pass a topic.
2. If a learner failed in a question and they already utilized their lifeline earlier in a topic, then another chance will not be given to the learner, and that skill will be treated as failed, which ultimately leads to failure of the topic.



3. There will be 1 lifeline/topic.

### **Trade-Offs between “Retry option” and “Lifeline option”**

If a learner failed in a question, I am planning to not give the retry option because of the following reasons:

- a. In the diagnostic test, we are aiming to judge the existing knowledge of a student not to enhance their learning. This is the reason we are omitting hints and solutions too.
- b. Example: In the case of MCQ, If a learner is confused between two options because of a missing skill and they eventually selected the right answer after multiple wrong answers. So using the retry option they can reach to the right answer but actually, they lack the corresponding skill, hence we will mark the first answer for evaluation and not provide the retry option.

In contrast, to make the test a little less strict, I am planning to introduce a lifeline option. This option saves learners from failing a topic after attempting a wrong answer.

Conclusion: Hence, the lifeline option is better in comparison to the retry option.

### **For the topic containing 1 diagnostic test skill**

1. Present 1 question successively.
2. If failed in any question, present another question from the same skill, but verify that the learner has not consumed their lifeline.
3. Else, proceed with the test.

### **For the topic containing 2 diagnostic test skills**

1. Present 2 questions successively.
2. If failed in any question, present another question from the same skill, but verify that the learner has not consumed their lifeline for that topic.
3. Else, proceed with the test.

### **For the topic containing 3 diagnostic test skills**

1. Present 3 questions successively.

2. If failed in any question, present another question from the same skill, but verify that the learner has not consumed their lifeline for that topic.
3. Else, proceed with the test.

Length of skill\_ids affecting the question count

Each skill in the diagnostic test should be passed individually in order to pass on a topic.

len(skill_ids)	# questions fetched	# questions presented (max)	Comments
1	2	2	<p>One question will be presented and if by any chance they consume their lifeline then the second question will be presented.</p> <p>The lifeline can be consumed for a skill, hence 2 questions are fetched from each skill.</p>
2	4	3	<p>Two questions will be presented and if by any chance they consume their lifeline then the third question will be presented.</p> <p>The lifeline can be consumed for any of the skills, hence 2 questions are fetched from each skill.</p>
3	6	4	<p>Three questions will be presented and if they consume their lifeline, then another question will be presented from the same skill.</p> <p>The lifeline can be consumed for any of the skills, hence 2 questions are fetched</p>

			from each skill.
--	--	--	------------------

Providing a “lifeline” and an “I don’t know” option

**As per discussion:** The learner needs an option to skip a question by the “I don’t know” option when they don’t understand the meaning of a question. Internally the “I don’t know” option will work similarly to the lifeline option i.e., present another question from the same skill, and skipping twice in a topic leads to marking that topic as failed.

### Diagnostic test algorithm

The algorithm provides the strategy to proceed with the next topic in the test after failing or passing any topic.

### Problem statement

In an Oppia classroom, there are multiple topics, and the learners should get a recommendation for getting started with a topic. For recommending a topic, a test should be taken (diagnostic test) which presents questions from multiple topics, and based on the performance 0 or 1 or 2 topics are recommended.

Design an algorithm for selecting topics from the topics DAG for the diagnostic test, and present it to the learners, and based on the performance on the earlier topics and current topic either the next topic is selected for the test or the current topic is recommended.

### Input:

- **topicsDag:** dict[str, List[str]]
  - Description: The input will be of type dict with topic id as key and a list of the immediate children(successors) topicIds as value.

### Output:

Based on the performance in the diagnostic test, a list of topicIds will be returned ( $0 \leq \text{length of the returned list} \leq 2$ ).

The list only contains the ids for those topics which are failed by the learner and for those topics, there should be no prerequisites that are untested or failed.

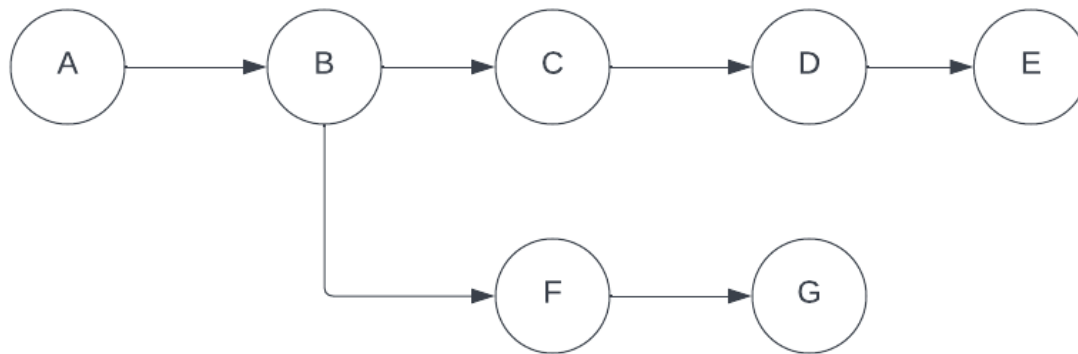
**Constraints:**

- The maximum number of nodes in the topics\_dag  $\leq 15$ .

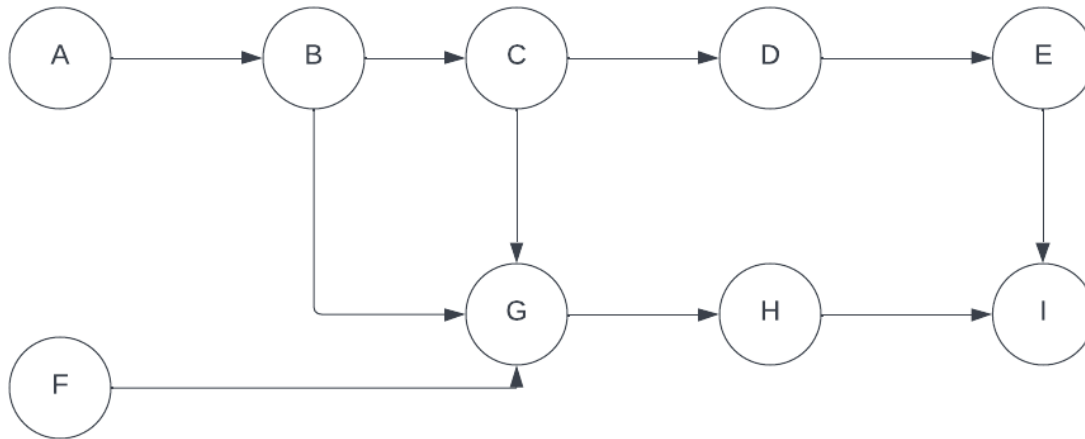
**Solution**

Sample DAG

**Example 1**



**Example 2**



From the adjacency list representation of a dag other data structures can be created like **topicIdToAncestorTopicIds** and **topicIdToSuccessorTopicId**.

- topicIdToAncestorTopicIds
  - A dict containing topic id as a key and listOfAncestorTopicIds as value.
  - **Example 1** (Following image 1)

```

topicIdToAncestorTopicIds = {
  'A': [],
  'B': ['A'],
  'C': ['A', 'B'],
  'D': ['A', 'B', 'C'],
  'E': ['A', 'B', 'C', 'D'],
  'F': ['A', 'B', 'C', 'D', 'E'],
  'G': ['A', 'B', 'C', 'D', 'E', 'F'],
}
  
```

- **Example 2** (Following image 2)

```
topicIdToAncestorTopicIds = {
  'A': [],
  'B': ['A'],
  'C': ['A', 'B'],
  'D': ['A', 'B', 'C'],
  'E': ['A', 'B', 'C', 'D'],
  'F': [],
  'G': ['A', 'B', 'C', 'F'],
  'H': ['A', 'B', 'C', 'F', 'G'],
  'I': ['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H']
}
```

- topicIdToSuccessorTopicIds
  - A dict containing topic id as key and listOfSuccessorTopicIds as value.
  - **Example 1** (Following image 1)

```
topicIdToSuccessorTopicIds = {
  'A': ['B', 'C', 'D', 'E', 'F', 'G'],
  'B': ['C', 'D', 'E', 'F', 'G'],
  'C': ['D', 'E'],
  'D': ['E'],
  'E': [],
  'F': ['G'],
  'G': []
}
```

- **Example 2** (Following image 2)

```
topicIdToSuccessorTopicIds = {
  'A': ['B', 'C', 'D', 'E', 'G', 'H', 'I'],
  'B': ['C', 'D', 'E', 'G', 'H', 'I'],
  'C': ['D', 'E', 'G', 'H', 'I'],
  'D': ['E', 'I'],
  'E': ['I'],
  'F': ['G', 'H', 'I'],
  'G': ['H', 'I'],
  'H': ['I'],
  'I': []
}
```

## Terminologies

- eligibleTopicIds: Those topicIds that are eligible to be picked on the next iteration.
- skippedTopicIds: Those topicIds that are skipped once a topic is passed or failed, they include ancestors or successors of a topic.
  - These are the topics that we have never tested & we don't want to test them because we have reached a conclusion based on earlier performance.
- passedTopicIds: Those topicIds that are passed by the learners.
- failedTopicIds: Those topicIds that are passed by the learners.

## Methods

- [Helper Functions](#)
- [Diagnostic Test Algorithm](#)

## External helper functions

1. **presentTopicForTest()**
  - Description: The method is used to present the questions from a list of diagnosticTestSkillIds associated with a topic. This function uses other services and modules in order to present the topics, but the scope of this doc is to only present the algorithmic part, hence the implementation of this method is abstracted.
  - Input: diagnosticTestSkillIds: A list of skill ids for the diagnostic test.

- Output: **testResult** and **numberOfAttemptedQuestions**.

Internal helper functions

**1. getTopicIdToAncestorTopicIdsDict()**

- Description: Input the topics DAG as adjacency list and modify it into [topicIdToAncestorTopicIds](#) dict

**2. getTopicIdToSuccessorTopicIdsDict()**

- Description: Input the topics DAG as adjacency list and returns the [topicIdToSuccessorTopicIds](#) dict.

Diagnostic Test Algorithm

**diagnosticTestAlgo()**

1. Initial conditions
  - a. totalNumberOfAttemptedQuestions = 0
  - b. eligibleTopicIds: List[str] = allTopicIds
  - c. skippedTopicIds: List[str] = []
  - d. passedTopicIds: List[str] = []
  - e. failedTopicIds: List[str] = []
2. Call **getTopicIdToAncestorTopicIdsDict()** and **getTopicIdToSuccessorTopicIdsDict()** method to create additional data structures from the adjacency list of the topics dag.
3. currentTopicId = find **next topicId** to test from eligibleTopicIds
  - a. Iterate on eligibleTopicIds and select a topic for testing based on the max(min(len(ancestors), successors)).
  - b. Follow [this](#) bookmark for complete reference.
4. **diagnosticTestSkillIds** = fetch diagnostic test skill ids from the currentTopicId.
  - a. **result, numberOfAttemptedQuestions = presentTopicForTest(diagnosticTestSkillIds)**
  - b. The result of the current topic is either pass or fail i.e, either the learner passed in the current topic or failed in the current topic.
  - c. numberOfAttemptedQuestions tracks the number of questions consumed to conclude whether the learner passed or failed in the current topic.
  - d. totalNumberOfAttemptedQuestions += numberOfAttemptedQuestions
5. If passed in the current topic (result === true)
  - a. **(Description: Passing a topic signifies the learner has the skills for the current topic as well as for all the ancestor topics.)**



- b. ancestorTopicIds = Get all the ancestor topic ids.
  - c. passedTopicIds += currentTopicId
  - d. eligibleTopicIds = eligibleTopicIds - (currentTopicId + ancestorTopicIds)
  - e. skippedTopicIds += ancestorTopicIds
- Else If failed in the current topic (result === false)
- f. successorTopicIds = Get all the successor topic ids.
  - g. failedTopicIds += currentTopicId
  - h. eligibleTopicIds = eligibleTopicIds- (currentTopicId + successorTopicIds)
  - i. skippedTopicIds += successorTopicIds
6. If [termination condition](#) reached, end the test. The conditions which checks whether the test has reached its ending point are as follows:
- a. len(failedTopicIds) == 0 and len(totalAttemptedQuestions) >= 15:
  - b. len(eligibleTopicIds) == 0 and len(failedTopicIds) > 0:
  - c. len(eligibleTopicIds) == 0 and len(skippedTopicIds) == 0
- Else If len(eligibleTopicIds) == 0 and len(skippedTopicIds) > 0 and len(totalAttemptedQuestions) < 15 and len(failedTopicIds) == 0:
- d. eligibleTopicIds = skippedTopicIds
  - e. skippedTopicIds = []
7. If the test ended:
- a. Create recommendedTopicIds list and add 1 or 2 topics for recommendation or the list will remain empty if the learner answers everything correctly.
  - b. The recommendation will be done based on the following table:

# topics recommended	len(failedTopicIds)	Description
0	0	Learner answered every question correctly.
1 or 2	> 0	Sort the failedTopicIds list in topological order. Then, recommend the first two root nodes (or the single

		root node, if there is only one).
--	--	-----------------------------------

### Edge Cases

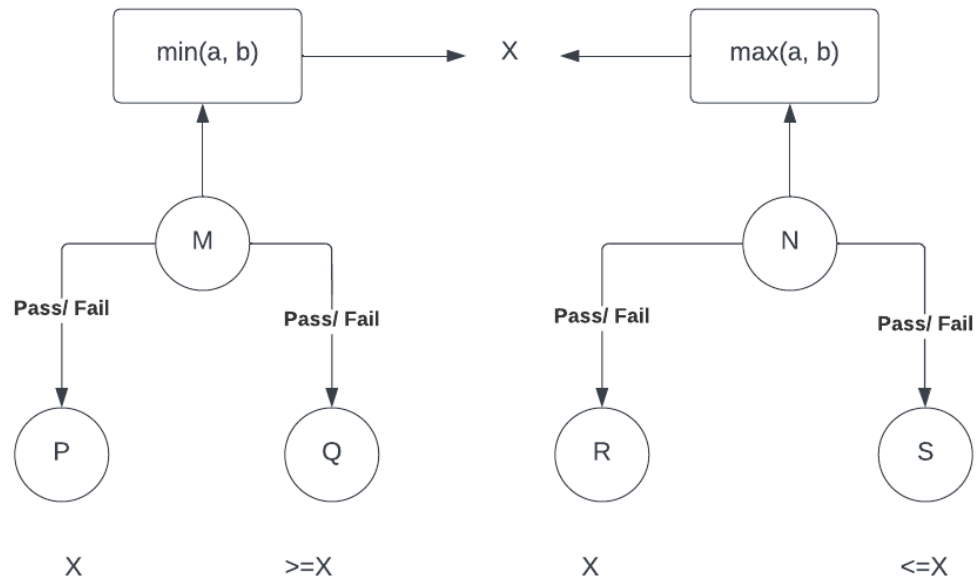
1. Let's assume the total number of attempted questions is 13 or 14. Now if the next topic is presented to the learner then there may be a chance to exceed the number of questions (15). In this case, we should complete the topic checking even, if 1 or 2 questions in the diagnostic test are extra.

### Selecting the next topicId from a list of eligibleTopicIds

#### Steps

1. For every topic to be selected from the eligibleTopicIds list, there are some topics that should be removed based on the result of the selected topic:
  - a. Some set of ancestors to be removed (If learner passed in the selected topic)
  - b. Some set of successors to be removed (If learner failed in the selected topic)
2. Judging criteria
  - [Practical consideration]:** Result of any topic i.e., Pass/Fail leads to removal of at least  $\min(a, b)$  topicIds from the eligible set.
  - [Optimistic condition]:** Result of any topic i.e., Pass/Fail leads to removal of the maximum number ( $\max(a, b)$ ) of topicIds from the eligible set.

Example:



Imaging two topics, M and N both are removing X number of topicIds from the eligible set based on different criteria:

- M removes the topicIds, based on  $\min(a, b)$  and N removes the topicIds, based on  $\max(a, b)$ .
- The optimal strategy to judge a topic is  $\min(a, b)$  because it ensures that in any case (pass/fail), X number of topicIds will always be removed from the eligible set.

3. Now for every topic to be selected, let's say **K** be a list containing  $\min(a, b)$  value for every topic id.
4. In order to reach the result in the shortest amount of time the topic which ensures the removal of the maximum number of topicIds after an iteration should be selected.
5. Thus for minimizing the length of eligibleTopicIds, the largest value should be selected from K i.e.,  $\max(k)$ .
6. Thus the algorithm to pick a node looks like a  $\max(\min(a, b))$

Terminating conditions for the diagnostic test.

**Convention**

- ☐ : Non-related
- ☐ : Break the loop.
- ☐ : Continue iteration

**Decision Table**

#	Length of eligibleTopicIds	Length of skippedTopicIds	Length of passedTopicIds	Length of failedTopicIds	Total attempted question	Conclusion
1	> 0	-	-	-	< 15	Continue
<p><u>Description:</u> Since the eligible set contains some topics ids, so continue the iteration for testing the topics.</p>						
2	> 0	-	-	0	>= 15	Break
<p><u>Description:</u> Eligible set contains some topicids, but the learner has attempted &gt;= 15 questions, without any failure so it is flagged to end the test.</p>						
3	0	> 0	-	0	< 15	Continue
<p><u>Description:</u> Eligible set is null and the learner has not attempted any wrong answer till now, so it is worth continuing the test and checking the knowledge from other topics which are skipped earlier (this is like the “second round” of the test). Thus making skippedTopicIds as eligibleTopicIds set.</p>						
4	> 0	> 0	-	> 0	-	Continue
<p><u>Description:</u> Eligible set is not null and the learner failed in at least one topic, in this case, the total number of questions attempted will be unrelated to the test because the test should reach the exact node for which the learner does not have the skills.</p>						

5	0	> 0	-	> 0		Break
<p><u>Description:</u> Eligible set is null and the learner has attempted a few wrong answers, so it is a good point for an early recommendation from the failedTopicIds set. (Note: we pick a failedTopicId to recommend that has no failed prerequisites.)</p> <p>It is impossible to have failed &gt; 0 and eligible &gt; 0 with the existing termination conditions. Hence the recommendedFailedTopicId will not have any prerequisites in the eligible set.</p>						
6	0	0	-	-	-	Break
<p><u>Description:</u> Eligible set &amp; skippedTopicIds set are null, so the test should end and recommendation is done from the failedTopicIds set (0 if the failedTopicIds set is empty).</p>						

Customizations required in existing question player to accommodate diagnostic test

## Components hierarchy

Color convention:

 : The component does not need any modification.

 : The component needs modification by accommodating some new customization arguments.

- Oppia-conversation-skin
  - Oppia-tutor-card
    - Oppia-audio-bar
    - Oppia-content-language-selector
    - Oppia-rte-output-display
    - Oppia-input-response-pair
  - Oppia-supplemental-card
    - Oppia-continue-button
  - Oppia-learner-answer-info-card
  - Oppia-progress-nav

- Oppia-continue-button
- Oppia-correctness-footer
- Oppia-ratings-and-recommendations
  - Oppia-feedback-popup
  - Oppia-ratings-display
  - Oppia-exploration-summary-tile
    - Oppia-learner-dashboard-icons

From the above hierarchy, it is shown that only 5 components need to accommodate new customization arguments.

Customization options for the components.

Component	Non-required functionality	Newly introduced arguments	Exploration player	Question player	Diagnostic test player
<b>Oppia-input-pair-response</b>  (This component is used only to show the input response pair.)	1. Feedback from the oppia should not be presented back to the learner.	1. <b>enableFeedback</b> : This argument will enable or disable the feedback for the answers i.e., the response part will not be shown if this argument is true.	enableFeedback = <b>True</b>	enableFeedback = <b>True</b>	enableFeedback = <b>False</b>
<b>Oppia-tutor-card</b>	1. The answer field should be removed after submitting the first answer.	1. <b>allowOnlySingleAttemptForAnswering</b> : This argument will be used to allow only a single attempt for each question i.e.	allowOnlySingleAttemptForAnswering = <b>False</b>	allowOnlySingleAttemptForAnswering = <b>False</b>	allowOnlySingleAttemptForAnswering = <b>True</b>

	<p>2. Multiple Input-responses should not be shown.</p> <ul style="list-style-type: none"> <li>○ The case when the learner wants to see all of its previous attempts for a question, should not be shown because in the diagnostic test they only get one opportunity to attempt any question.</li> </ul>	<p>remove the answer form field after the first attempt.</p> <p>2. <b>showOnlyLastInputPairResponse</b>: This argument will be used to disable the feature that shows, multiple input response pairs for the previous attempts.</p> <p>3. <b>enableFeedback</b>: This argument will be required by the oppia-input-pair-response component.</p>	<p>showOnlyLastInputPairResponse = <b>False</b></p> <p>enableFeedback = <b>True</b></p>	<p>showOnlyLastInputPairResponse = <b>False</b></p> <p>enableFeedback = <b>True</b></p>	<p>showOnlyLastInputPairResponse = <b>True</b></p> <p>enableFeedback = <b>False</b></p>
<p><b>Oppia-supplemental-card</b></p> <p>(This component is used to present the HTML data for those interactions which are not inline like</p>	<p>1. Oppia Feedback for wrong / right answers should be disabled.</p>	<p>1. <b>enableFeedback</b>: This boolean flag will be used for disabling the feedback for both the cases i.e., for the wrong attempt or for the right attempt of any question.</p>	<p>enableFeedback = <b>True</b></p>	<p>enableFeedback = <b>True</b></p>	<p>enableFeedback = <b>False</b></p>

<p>“Drag And Drop”. This component also shows the feedback for wrong/right answers.)</p>					
<p><b>Oppia-progress-navigation</b></p>	<p>1. Card migration: Forward and backward movement of previous questions and current questions should not be disabled.</p>	<p>1. <b>enableNavigationThroughCardHistory</b>: This argument will enable or disable the forward and backward movement between the cards.</p>	<p>enableNavigationThroughCardHistory = <b>True</b></p>	<p>enableNavigationThroughCardHistory = <b>True</b></p>	<p>enableNavigationThroughCardHistory = <b>False</b></p>
<p><b>Conversation skin component</b></p>		<p>Based on the customization arguments of the above components, this root level component can be structured in two ways:</p> <ol style="list-style-type: none"> <li>a. All of the above customization arguments can be passed directly to this component and from here they were passed into descendant components.</li> <li>b. A single field can be passed, like “isDiagnosticTestPlayer”. And from here all the necessary customizations will be passed into the</li> </ol>			



		<p>respective components.</p> <p><b>Conclusion:</b> As per discussion the conversation skin component should take all the arguments as inputs that were required by the descendant components.</p> <p>The list of new arguments are</p> <ol style="list-style-type: none"> <li>1. enableFeedback</li> <li>2. allowSingleAttemptOfAnswering</li> <li>3. showOnlyLastInputPairResponse</li> <li>4. enableNavigationThroughCardHistory</li> </ol>			
--	--	--	--	--	--

## Implementation Approach

Convention:

[A]: Adds new method.

[U]: Updates any existing method.

### Topic Editors

Topic editors can potentially be able to select and reorder skills inside the topic editor page.

[Backend Changes]

#### 1. Model Layer

- a. [U]: Add a new field, **diagnostic\_test\_skill\_ids** to the topic model class.

## 2. Domain Layer

- a. [U]: Add the **diagnostic\_test\_skill\_ids** field among all the topic-related classes in the domain layer. Example: Topic class, `_create_topic()` method, etc.
- b. Add command in topic domain
  - Add a command to update the diagnostic test skill ids in the datastore. This command reflects the changes made by the topic editors.
  - Name: `CMD_UPDATE_DIAGNOSTIC_TEST_SKILL_IDS`
    - Required fields: `diagnostic_test_skill_ids`
- c. [A]: `update_diagnostic_test_skills()`
  - Method to handle the model layer changes based on the command `CMD_UPDATE_DIAGNOSTIC_TEST_SKILL_IDS`.

## 3. Controller Layer

Note: As per the current requirement no need to write any new handlers. The cases which will be covered are as follows:

- Gets skill description from the list of skill ids: These skill descriptions will be fetched by using the short skill summary class in the frontend.

## [Frontend Changes]

### 1. Domain

- Create command `CMD_UPDATE_DIAGNOSTIC_TEST_SKILL_IDS` (similar to the backend).
- [U]: Topic domain object
  - Add a new field **diagnostic\_test\_skill\_ids**, in the `TopicBackendDict`.

### 2. Component / Services / Backend API

- [A]: `fetchListOfSkillDescription`
  - Iterate over all the subtopics and extract skill descriptions from the short skill summary (Add file location).
    1. `getSkillSummary` in `subtopic.model.ts`
- [A]: `updateDiagnosticTestSkills` (`topic-update-services.ts`)
  - Update and save the topic object with the new field **diagnosticTestSkillIds**.

### 3. HTML

- Add a card (md-card) in topic-editor-tab.directive.html.
  - This card will contain the list of skills present inside the topics (topic.subtopics)

#### [Special Cases]

1. Disable publish button and save button and present an error if none of the skills were added for the diagnostic test.
2. Remove the “Add diagnostic test skills” if the topic editor already added 3 skills for the diagnostic test.
3. If any skill gets deleted from the topic and skill dashboard or any skill is removed from a topic, then the diagnostic\_tes\_skill\_ids should be updated appropriately.

## Curriculum admins

The curriculum admins can potentially be able to input the dependency tree between the topics from the classroom admin page. The dependency tree is in the form of a dict with topic name as key and parent topic name as value.

## Classroom Config section

### Backend

- Model
  - A new **ClassroomConfigModel** should be created for storing and fetching the classroom-related data like url\_fragment, topic\_ids, etc. Currently, the classroom config data is stored in config\_model which is handled by AdminHandler, but in order to restructure the data flow, this model should be used.
  - Fields: The schemas of these fields are given [here](#).
    - Classroom name (indexed)
    - Classroom\_url\_fragment
    - Course\_details
    - Topic\_list\_intro
    - topic\_ids
- Domain
  - **[U]**: Classroom\_domain: The file is already present in the domain layer which should be modified to contain the analogous domain class for the ClassroomConfigModel.

- [U]: Classroom\_service: The file is already present in the domain layer which should be modified to contain all the helper functions related to classroom config models.
- Controllers
  - [A]: ClassroomConfigHandler: A handler class for communicating with the frontend.
    - Methods:
      - def get()
        - **Decorator:** can\_access\_admin\_page
        - Returns the classroom\_config\_properties for populating them in the frontend, so that curriculum admins can edit them.
      - def put()
        - **Decorator:** can\_access\_admin\_page
        - Updates the changes which were done by the curriculum admin.
        - **Payload:** classroom\_config\_dict.

[Frontend]

- Classroom-config component: This component presents the classroom-config card from which the curriculum admins can edit the properties.
  - Methods:
    - [A]: getClassroomConfigProp()
    - [A]: updateClassroomConfigProp()

## Diagnostic Test section

[Backend]

- Model
  - The properties of the **DiagnosticTestModel** class are described [above](#).
- Domain
  - Diagnostic test domain: A file should be created in the domain layer, which should contain the domain class for the diagnostic test model.
  - Diagnostic test service: A file should be created in the domain layer, which should contain the helper functions related to the diagnostic test models.

- Controllers
  - [A]: DiagnosticTestTopicHandler
    - def get()
      - Decorator: open\_access
      - URL query param: classroom\_url\_fragment.
      - Fetches the topics\_DAG from the diagnostic test model.
      - Returns the topics\_adjacency\_list to the frontend.
    - def post()
      - Decorator: can\_access\_admin\_page
      - URL query param: classroom\_url\_fragment.
      - Payload
        - Updated topics\_adjacency\_list dict.

#### [Frontend]

- Diagnostic-test-admin component: This component present the diagnostic test card in the classroom admin page. This component provides the opportunity to either edit or create the topics DAG.
  - Methods:
    - getTopicsDag()
    - updateTopicsDag()

#### [Common Backend Changes]

##### Controllers Layer

- [A]: ClassroomAdminPage
  - def get()
    - Decorator: can\_access\_admin\_page
    - [A]: Render template classroom-admin-page.mainpage.html

#### [Special Cases]

1. Delete the topic name from the diagnostic test model, in the case when a topic gets deleted from the topic and skill dashboard.
  - a. Remove the topic node from the diagnostic test model if a topic gets deleted.

- b. Example approach:
- i. M, N, O, and P are four arbitrary topics present in a classroom.
  - ii. M is the parent of N.
  - iii. N is the parent of O & P.
  - iv. If N gets deleted, from the topic & skill dashboard:
    1. Link O & P to M as its immediate children. Now M is the parent of O & P.

## Learners

### Backend

- Model
  - A **DiagnosticTestModel** should be created for the retrieval of the topics DAG by the diagnostic test interface so that questions presented in the diagnostic test are related to these topics.
  - The description of the **DiagnosticTestModel** is given [above](#).
- Domain
  - *Implemented in Curriculum admin part.*
- Controllers
  - **[A]**: DiagnosticTestPageHandler
    - def get()
      - **[A]**: Renders the main HTML page for the diagnostic test.
  - **[A]**: DiagnosticTestDataHandler
    - def get()
      - URL query param: classroom-url-fragment.
      - Decorator: **Open access**. Currently, we allow everyone to play the topics on the classroom page. Thus if a learner is allowed to play the topics with or without sign-in, they should be able to give the diagnostic test with or without sign-in.
      - Returns the diagnostic test topics DAG from the model by classroom-url-fragment.

### Frontend

- Diagnostic-test-backend-api-service
  - Interacts with the backend and makes HTTP calls.

- Methods
  - getTopicsDagAsync()
  - updateTopicsDagAsync()
- Diagnostic-test-algorithm-services
  - Fields
    - **recommendedTopicIds**
    - topicIdToAncestorTopicIds
    - topicIdToSuccessorTopicIds
    - passedTopicIds
    - failedTopicIds
    - skippedTopicIds
    - eligibleTopicIds
  - Methods
    - initialize(topicsDag)
      - Initializes all the fields.
    - getNextSkillIdsForDiagnosticTest()
      - Pick the next topic from the eligible set.
      - Get skillIds for the current topicId
      - Return the skillIds
      - (Step 3 and 4 from the algorithm)
    - recordTopicPassed() / recordTopicFailed()
      - These methods are called when the diagnostic test skills from the currentTopicId are tested.
      - Updates the diagnostic test state data i.e., passedTopicIds, failedTopicIds, etc, and other parameters based on the result of the current topic test.
      - (Step 5 to 7 from the algorithm)
    - getRecommendedTopicIds()
      - This method is called when the diagnostic test will be ended.
      - Return the recommendedTopicIds that were created in the algorithm.

## [Backend Changes]

### 1. Controller Layer

- [A]: DiagnosticTestPageHandler
  - def get()
    - [A]: Renders the main HTML page for the diagnostic test.
- [A]: DiagnosticTestDataHandler
  - def get()
    - Returns the diagnostic test topic dependency tree.

## [Frontend Changes]

1. [A]: questionPresentationAlgo()
  - (Discussed above)

## [Special Cases]

1. First-time learners should get a popover for the diagnostic test feature.
  - a. The first-time interaction can be traced by using local-storage.service inside the classroom-page component.
    - A method should be written in the local-storage.service and that corresponding method will be used inside the ngOnInit method of the classroom-page.component.ts
2. The popover should disappear automatically after 2 mins.
  - a. Auto disappear can be provided by using a javascript function **setTimeout(function, time)**
  - b. The time should be 120000 ms (or 2 mins).
3. Provide a "I don't know" option to the learner for the questions in which they don't have any idea on how to attempt a particular question.



# Testing Plan

## E2e testing plan

#	Test name	Initial setup step	Step	Expectation
1	Learner	Login / without login	Visit the math classroom page	The page should load.
			A popover was presented to the learner, featuring the diagnostic test.	The popover should be presented.
			Click the diagnostic test button and navigate to the diagnostic test page.	The page should load.
			Give answers to the questions and complete the test.	Should be able to get the topic recommendation.

## Feature testing

Does this feature include non-trivial user-facing changes?

**YES**

# Implementation Plan

## Milestone Table

No.	Description of PR / action	Prereq PR numbers	Target date for PR creation	Target date for PR to be merged
<b>Milestone 1</b> Curriculum admins should be able to use a “classroom administration” page to configure details of each classroom, enter the dependencies between topics in that classroom, and enter details for the diagnostic test. A comprehensive suite of backend integration test cases that convincingly should demonstrate that the recommendation system entered by the curriculum admin works correctly.				
1	Update topic model and its related domain methods for accommodating diagnostic test skills.	None	6-July	16-July
2	Provide functionality to add diagnostic test skills to a topic from the topic editor page	1	12-July	22-July
3	Add a diagnostic test model class.	None	16 July	26-July
4	Add domain layer functionalities for the diagnostic test model.	3	21 July	31-July
5	Add handlers and backend APIs for the diagnostic test	4	26-July	05-August
6	Add classroom config models and update existing related domain methods	None	31-July	11-August

7	Add controllers and backend apis for the classroom config.	6	05-August	15-August
8	Create classroom admin page and add classroom config card.	7, 5	12-August	22-August
9	Add topics dependency input feature on the classroom admin page	8	22 August	2-Sept
	Buffer time to fix any reported bugs			
<p>Milestone 2</p> <p>Learners should be able to visit the Math Classroom page and take an adaptive diagnostic test that surfaces 0, 1 or 2 topic recommendations for them to pursue.</p>				
10	Add controller layer handlers in the backend to visit the diagnostic test player page.	None	15-Sept	22-Sept
11	Add enableFeedback, allowOnlySingleAttemptForAnswering, and showOnlyLastInputPairResponse customization arguments for the tutor card component and enableFeedback argument for the input-pair response component.	None	21-Sept	28-Sept
12	Add enableFeedback customization argument for the supplemental card component and enableNavigationThroughCardHistory argument for the progress nav component.	None	27-Sept	5-Oct

13	Add enableFeedback, allowOnlySingleAttemptForAnswering, showOnlyLastInputPairResponse, and enableNavigationThroughCardHistory customization arguments for the conversation skin component.	None	4-Oct	15-Oct
14	Add diagnostic test player page component and service.	11, 12, 13	12-Oct	22-Oct
15	Add diagnostic test result page component.	14	16-Oct	21-Oct
16	Add functionality to end diagnostic test and present result page.	15	20-Oct	25-Oct
17	Add the "diagnostic test" button on the classroom page and a popup introducing the diagnostic test functionality to the new learners.	16	27-Oct	3-Nov
18	Add E2E for the diagnostic test functionality.	17	5-Nov	12-Nov
	Buffer time to fix reported bugs			

## Future Work

The diagnostic test feature should be used for story recommendations inside a topic.