

Section 1: About You

Goals and Objectives

Why do you want to do a GSoC project with Oppia?	Open-source projects like Oppia thrive on community contributions. Participating in an Oppia GSoC project allows individuals to contribute to open education and collaboration goals by working on a freely available project	
What do you hope to learn/achieve during GSoC?	I want to gain real-world experience, contribute to open-source projects, learn new technologies & skills and network with peers and mentors.	
Which Oppia teams have you collaborated with, and what have you done on those teams?	LaCE Team	In the future, I will assist other team members and new contributors because I was recently a new contributor.
Contact information	Email: pritsingh2317@gmail.com	
Preferred method of communication	Google Chat Google meet	
Which timezone will you primarily be in during the summer?	IST	
(If you are a student) When are your school holidays?	NA	
What other obligations might you need to work around during the summer?	No other obligations	
Planned time commitment	How much time do you plan to commit to this project (per week or per day) between May 27 and Aug 19? Be sure to provide a realistic estimate and be deliberate about this. E.g. your schedule shouldn't involve working continuously (e.g. 7 days a week), since this is unlikely to lead to a healthy balance or a good experience.	
	I aim to dedicate 25-30 hours per week, allocating time across 5 days in a week, to fully engage with this project.	

Section 2: About Your Project

Project Details

Project title <i>(should match the one on the Project Ideas list)</i>	Redesign the topic page
Project size <i>(should match the options on the Project Ideas list)</i>	Medium (~175 hours)
Why did you choose this project?	This project includes full stack knowledge. I want to enhance my skill in full stack development and gain practical experience through involvement in real-world projects.

Required Skills

WEB	
I can write Python code with unit tests. ¹	https://github.com/oppia/oppia/pull/19241 https://github.com/oppia/oppia/pull/20116

¹ To develop this skill: Some [LaCE](#) and [Contributor Dashboard](#) issues have a backend component, and many [Dev Workflow](#) issues do too. Focus on issues that aren't marked as "backlog".

I can write TS + Angular code with unit tests. ²	https://github.com/oppia/oppia/pull/19762 https://github.com/oppia/oppia/pull/20116
I have good UI judgment and attention to detail. ³	https://github.com/oppia/oppia/pull/19372 .
I can debug and fix CI failures / flakes. ⁴	https://github.com/oppia/oppia/pull/19762
I can write or modify e2e or acceptance tests. ⁵	Replace this text with links to PR(s) that demonstrate that you can write or modify e2e/acceptance tests.
I can communicate effectively using debugging docs . ⁶	Replace this text with links to debugging docs you have written that exhibit clear communication and that resulted in the issue eventually being solved.
I can write or modify Beam jobs . ⁷	N/A
I have participated in QA testing. ⁸	Participated in release testing :V3.3.5 Participated in release testing: V.3.3.6

Project Timeframe

Note: Oppia will only be offering a single GSoC coding period timeframe this year, starting on **May 27**. All work for Milestone 1 must be completed and submitted by **Jun 28** for internal feedback (with any revisions due by **Jul 8**), and all work for Milestone 2 must be completed and submitted by **Aug 12** for internal feedback (with any revisions due by **Aug 19**). We will not be able to extend these deadlines.

Coding period	<ul style="list-style-type: none"> I will adhere to the above deadlines.
---------------	---

Communication Channels

Note: The Oppia team places a high emphasis on communication, and we have found that daily contact between contributors and mentors is important for helping keep projects on track. This is why we ask that contributors send short daily updates to their mentors explaining what they have done, where they are stuck, and what they plan to do next.

I can commit to sending daily updates to my mentor by email, each day I work during the GSoC period.	<ul style="list-style-type: none"> Yes
In addition to the above: how often, and through which channel(s), do you plan on communicating with your mentor?	I plan on communicating with my mentor at least two or three times a week, through Google chat and Google meet.

Section 3: Proposal Details

Problem Statement

Target Audience	<ul style="list-style-type: none"> Primary audience: Learners Secondary audience: Teachers and Parents who may be utilizing the content on the site to educate the learners
Core User Need	<ul style="list-style-type: none"> As a learner, I want to be able to learn and practice different lessons through a topic-viewer page.
What goals do we want the solution to achieve?	<ul style="list-style-type: none"> The goal of this project is to enhance the design of the topic page, providing a cohesive view of lessons, practice sessions, and links to revision cards (subtopic pages), enabling learners to practice the skills acquired in the respective lessons.

Section 3.1: WHAT

Key User Stories and Tasks that will be implemented

#	Title	User Story Description	Story	Priority ⁹	List of tasks needed to	Links to mocks /	Task success
---	-------	------------------------	-------	-----------------------	-------------------------	------------------	--------------

² To develop this skill: Most [LaCE](#) and [Contributor Dashboard](#) issues have a frontend component. Focus on issues that aren't marked as "backlog".

³ To develop this skill: Tackle a non-backlog responsiveness issue from the [LaCE team](#).

⁴ To develop this skill: See issues labelled "[CI breakage](#)" on GitHub, or look at [CI failures in develop](#) (or cross-signs [here](#)) and figure out how to fix them.

⁵ To develop this skill: Solve part of issue [#17712](#) by covering the CUIs for one or more sets of users.

⁶ To develop this skill: Tackle an issue that requires creating a debugging doc that you share with the broader team. Fixing CI failures counts.

⁷ To develop this skill: See [this doc](#) for some examples of issues to work on.

⁸ To develop this skill: Join the release testers mailing list at oppia-release-testers@googlegroups.com, and look out for calls to help with release testing.

⁹ Use the **MoSCow** system ("Must have", "Should have", "Could have"). You can read more [here](#).

		(role, goal, benefit-to-user) "As a ..., I need ..., so that ..."	Success		achieve the goal (this is the "User Flow")	prototypes, and/or sections spec'ing ¹⁰ out the task's user flows and product requirements. ¹¹	criterion
1.	Understanding Topic Page	As a learner who has accessed a page of a Topic for the first time, I need to understand what that topic is about, so that I can decide if that topic is right for me.	Learner reads relevant information on Topic Page	Must have	Access Topic Page Read introduction about the topic Decide if the topic is right for them	High-Fi	Read relevant information about topic
2.	Starting first lesson	As a learner who has already decided which Math topic I want to study, I need to check the lessons that Oppia offers for that topic, so that I can choose a lesson and start learning/improving my skills in Math.	Learner starts a lesson from the Topic Page	Must have	Access Topic Page See list of available chapters/lessons Choose a chapter See a clear CTA to start the lesson Start a lesson	High-Fi	Click on "start" to start lesson
3.	Lesson unavailability in the website language	As a learner who has chosen a language other than English for the website, I want to know if the lesson's not available in my language so that I don't waste time doing it.	Learner is aware that the lesson is not available in the preferred language in advance	Must have	Access oppia.org Select preferred language (e.g. Brazilian Portuguese or PT-BR) Access Topic Page See the list of chapters/lessons See that the lesson is not available in the preferred language (e.g. PT-BR)	Requirements High-Fi	View unavailability of the lesson in the preferred language (voice over and text)
4.	Secondary language selection	As a learner who has identified that a lesson is not available in my preferred language and is familiar with another language, I need to be able to choose another language for that lesson, so that I can start learning in a language that I understand.	Learner selects a secondary language for the lesson	Must have	- Access Topic Page - See the list of chapters/lessons - See that the lesson is not available in the preferred language (e.g. PT-BR) - Select a secondary language (e.g. Spanish) - Start the lesson in the secondary language - Access Topic Page - See the list of chapters/lessons - Click to start lesson - View dialog box - Select a secondary language (e.g. Spanish) - Start the lesson in the secondary language	Requirements High-Fi	Select secondary language
5.	See status of previously started lessons	As a logged-in learner who has previously started a lesson, but hasn't finished it, and now is returning to the topic page, I want to see the status of progress, so that I can identify which lesson is in-progress and resume it.	The learner views the progress and clicks on "Resume"	Must have	Access Topic Page See the list of chapters/lessons See status of progress bar of each chapter/lesson	High-Fi	View progress bar
6.	Resuming a lesson	As a logged-in learner who has identified an uncompleted lesson in the topic page, I want to resume that lesson, so that I can complete it.	Learner views the previously started lesson and resumes it.	Must have	See uncompleted progress bar Click on "Resume"	High-Fi	Click on "Resume" to resume the previously started lesson
7.	Viewing	As a logged-in learner	The	Could	See completed progress	High-Fi	Click to review

¹⁰ Specs are important if the user journey has multiple cases (e.g. error types for wrong answers; different choices a user can make; etc.) and these need detailed enumeration.

¹¹ If mocks/prototypes have been validated by e.g. user studies, please provide details. This can help increase confidence in the solution, though it's typically only needed in cases when we're unsure about key assumptions, or there are risks that the solution won't resonate well with users.

	and reviewing completed lesson	who has identified a completed lesson in the topic page, I want to review the story together with the correct answers, so that I can remember what I've learned in the lesson.	learner sees the story and the correct submitted answers.	Have	bar		the lesson
					Click on CTA to view the lesson		
					See dialog asking to confirm if they would like to start over the lesson or review		
					Confirm that want to review the lesson		
					Review the lesson/story with correct answers		
8.	Starting over a previously completed lesson	As a logged-in learner who has completed a lesson a long time ago and forgot everything about it, I want to retake the lesson from the beginning, so that I can be reintroduced to the Math concepts.	Learner retakes a previously done lesson	Could have	Access Topic Page	Github issue	Click to start over the lesson
					See the list of chapters/lessons	High-Fi	
					See completed progress bar		
					Click on CTA to view the lesson		
					See dialog asking to confirm if they would like to start over the lesson or review		
					Confirm that want to start over the lesson		
					Start over the lesson		
9.	Representa tion of Chapter Availability	As a learner, I want to identify available chapters, so that I'm aware how far I can progress in learning this topic currently and how much more content can I anticipate in the future	Learner completes all available chapters, does not waste time attemptin g to open/click on unavailabl e chapters and anticipate s further chapter launches in the future	Could have	- View list of 'coming soon' chapters in Story Card - Attempt all available chapters - Optionally, subscribes to email notifications for future chapter launches	Requirements High-Fi	Completes all available chapters and does not waste time attempting to open/click on unavailable chapters
10.	Reviewing skill at the end of a set of lessons	As a learner who has just completed a set of lessons, I need to review the skills I've just learned, so that I can solidify this knowledge before practicing.	Learner checks review card of the skills introduced in the previous "N" lessons	Should have	Start lessons	High-Fi	Click on CTA to check review card of introduced skills
					Complete a set of "N" lessons that fully introduce new skills	Note: It might need more than 1 lesson to introduce a group of skills within a skill, that's why the suggestion to check the review card will be after a set of "N" lessons, and not after the end of every lesson.	
					At the end of the practice session, see a screen with the information that they have finished learning a new skill		
					Check review card of the introduced skills		
11.	Practice a skill at the end of a set of lessons	As a learner who has just completed a set of lessons, I need to practice the skills I've just learned, so that I can solidify this knowledge.	Learner starts a practice session of the skills introduced in the previous "N" lessons	Should have	Start lessons	Mock	Click on CTA to practice introduced skills
					Complete a set of "N" lessons that fully introduce new skills	Note: It might need more than 1 lesson to introduce a group of skills within a skill, that's why the suggestion to practice will be after a set of "N"lessons, and not after the end of every lesson.	
					At the end of the practice session, see a screen with the information that they have finished learning a new skill		
					View skill introduced in the "N" previous lessons		
					Start a practice session of the skill		

12.	Review in another study session	As a learner who has completed one or more lessons previously, and now is returning to the platform for another study session, I want to review the previously learned skill, so that I remember what I've learned previously.	Learner checks the review card of a previously learned skill	Must have	Access Topic Page	High-Fi	Check study card/review card
					See the list of chapters/lessons		
					See skills related to the lessons		
					Click on CTA study the skill		
13.	Practice in another study session	As a learner who has completed one or more lessons previously, and now is returning to the platform for another study session, I want to practice the previously learned skill, so that I can solidify this knowledge.	Learner starts a practice session of the skill introduced in the previously done lessons	Must have	Access Topic Page	High-Fi	Click on review card
					See the list of chapters/lessons		
					See skills related to the lessons		
					See option to practice		
					Click on CTA to practice the skill		
14.	Seeing concepts within a skill	As a learner who wants to start a practice session, I need to know more details of the skill, so that I am aware of what I'm learning.	Learner checks details of the skill	Could have	Access Topic Page	High-Fi	Click on CTA to view skills within the skill.
					See list of skills		
					See CTA to know more		
					See list of concepts within a skill		
15.	Start over uncompleted lesson	As a learner who started a lesson some time ago and doesn't remember the lesson story, I need to be able to start over the lesson, so that I see the story from the beginning.	Learner starts over previously completed lesson	Should have	Access Topic Page	High-Fi	Click to start over previously completed lesson
					View previously completed lesson		
					Click on resume		
					View dialog box		
					Click on "start over"		
					Star over lesson		
16 ..	View skill progress	As a logged-in learner who is doing practice sessions, I need to view my progress in the skill, so that I'm aware if I'm learning that content or not.	View skill mastery	Should have	Access Learner Dashboard See skill mastery Or Access Topic Page View skill View skill mastery	High-Fi	View skill mastery

Technical Requirements

Additions/Changes to Web Server Endpoint Contracts

#	Endpoint URL	Request type (GET, POST, etc.)	New / Existing	Description of the request/response contract (and, if applicable, how it's different from the previous one)
1.	/explore/<exp_id>?suf=<story_url_fragment>&tuf=<topic_url_fragment>&cuf=<classroom_url_fragment>&nid=<node_id>&tlc=<translation_language_code>&vlc=<voiceover_language_code>	GET	Existing	This URL is used to access the exploration viewer page with specific translation and voiceover languages. These languages are set as query parameters in the URL and sent to the exploration viewer page for translation.
2	/get_exploration_translation_and_voiceover_languages/<exp_id>	GET	New	<p>The API will be used to fetch translation & voiceover languages for exploration. The API will trigger the ExplorationLanguageHandler class.</p> <p>Request : exploration_Id</p> <p>Response: list of translation_language_codes and voiceover_language_codes</p> <p>Data structure :{ 'translation_language_codes': [...], 'voiceover_language_codes': [...] }</p>

3.	/explore/<exp_id>?suf=<story_url_fragment>&tuf=<topic_url_fragment>&cuf=<classroom_url_fragment>&nid=<node_id>&review_lesson="true"	GET	Existing	This URL takes 'review_lesson' as a query parameter, which is triggered when the learner clicks the review button then the exploration viewer page will be displayed from the starting state card along with the saved progress.
4.	/topic_data_handler/<classroom_url_fragment>/<topic_url_fragment>	GET	Existing	A new field 'preferred_site_language_code' will be added in the TopicPageDataHandler to fetch the site language of the user.
5.	/get_user_saved_checkpoints?exploration_id=<exploration_id>	GET	Existing	The new handler will be created named 'UesrSavedCheckpoints' which is triggered by this URL where exploration_id passed as a query parameter to fetch the user checkpoint progress.

Calls to Web Server Endpoints

#	User	Endpoint URL	Request type (GET, POST, etc.)	Description of why the new call is needed, or why the changes to an existing call is needed
1.	Learner	/learn/<classroom_url_fragment>/<topic_url_fragment>/story	GET	In the HTML of this URL, a feature flag will be introduced. If it is true, the new UI for the topic viewer page will be displayed. If it is disabled, then the current UI will be displayed. Mocks Related to topic viewer page
		/explore/<exp_id>?suf=<story_url_fragment>&tuf=<topic_url_fragment>&cuf=<classroom_url_fragment>&nid=<node_id>&tlc=<translation_language_code>&vlc=<voiceover_language_code>	GET	This URL is used to access the exploration viewer page with specific translation and voiceover languages. These languages are set as query parameters in the URL and sent to the exploration viewer page for translation.
		/get_exploration_translation_and_voiceover_languages/<exploration_id>	GET	The API will be used to fetch translation & voiceover languages for exploration. The API will trigger the ExplorationLanguageHandler class. Request : exploration_Id Response: list of translation_language_codes and voiceover_language_codes Data structure :{ 'translation_language_codes': [...], 'voiceover_language_codes': [...] }
		/topic_data_handler/<classroom_url_fragment>/<topic_url_fragment>	GET	A new field 'exploration_preferred_language_code' will be added in the TopicPageDataHandler to fetch the Exploration preference language of the user

UI Screens/Components

#	ID	Description of new UI component	i18n required ?	Mock/spec links	A11y requirements
1.	Topic viewer page	Topic viewer page is visible to learner	Yes	Topic viewer Page	<ul style="list-style-type: none">• Yes<ul style="list-style-type: none">○ Use semantic HTML elements(<header>,<main>,<nav>)etc to structure the content.○ Use all the interactive elements(links, buttons, form controls) are accessible via keyboard○ I'll provide descriptive alt text for all images.○ Text and interactive elements have sufficient color contrast.

					<div>Lighthouse accessibility tests to maintain at a 1.0 score:</div> <div><pre>module.exports = { numberOfRuns: 3, puppeteerScript: 'puppeteer-login-script.js', urlShards: { 1: ['http://localhost:8181/ topic_editor'], 2: ['http://localhost:8181/learn/math/dummy-to-pic-one'], }, }</pre></div>
2.	Topic Editor page	Topic editor page is not visible to learner	No	Topic editor page	<div><ul style="list-style-type: none">No</div>

Data Handling and Privacy

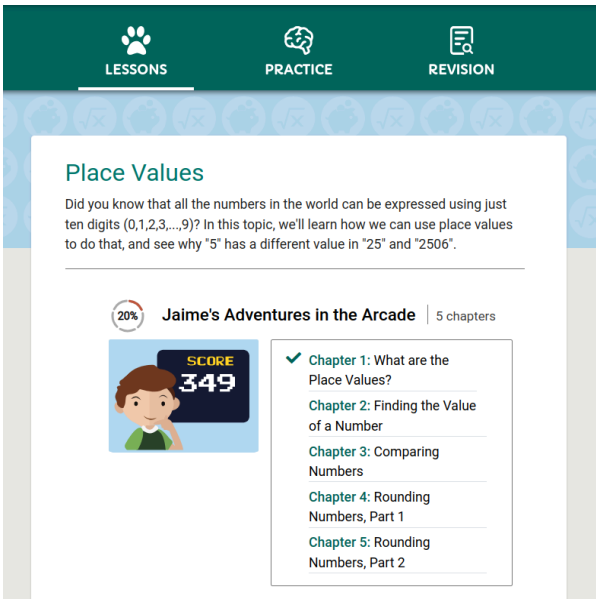
NA

Section 3.2: HOW

Existing Status Quo

Currently in the topic viewer page, there are 3 tabs

- Story
 - This tab lists all the lessons present in a topic.
- Practice
 - This tab lists all the subtopics of the topic. Selecting one or more subtopics, the user can start the practice session.
- Revision
 - This tab lists all the review cards.
- Languages for translations and voiceovers that learners may want to use are only available after they start the lesson.




Solution Overview

1. Update the design of the topic page to present a more integrated view of the lessons, practice sessions, and links to revision cards (subtopic pages), so that learners can practice the skills they learned in the relevant lesson.
2. New design incorporates the selections of translation language and voiceover languages within the topic page itself and follows the overall site language as a default.

Place Values

Did you know that all possible numbers of things can be expressed using just ten digits (0, 1, 2, 3, ... , 9)? In this topic, we'll learn how we can use place values to do that, and, for example, how "5" has a different value in "25" and "2506".

Story: "Help Jaime win the Arcade Game"




In this story, we'll follow Jaime and his sister Nic as they learn how to represent and read the value of a number. Start with lesson 1!

☒ Show lessons

☒ Show practice

View skills




Lesson 1: What are place values?
Jaime learns the place value of each digit in a big number.

Start

Practice 1

Practice the skills you've learned in lesson 1.



Skill 1

The Place Values and Their Names

Study

Practice

Third-Party Libraries

No.	Third-party library name and version	Link to third-party library	Why it is needed	License ¹² (if third-party library)	[Android only] Min / target / max SDK version that the library supports
1	N/A				

“Service” Dependencies

No.	Dependency name	Why it is needed	What our plan is, if the dependency fails under us
1	N/A		

Impact on Other Oppia Teams

NA

Solution Overview

Project Objectives:

- Revise the design of the topic page to provide a cohesive overview.
- Integrate lessons, practice sessions, and links to revision cards (subtopic pages) for enhanced accessibility.
- Facilitate more effective practice of acquired skills within the corresponding lesson.

Additional Features to Address Learner Concerns:

- Embed language selection options directly within the topic page interface.
- Default to the preferred language to mitigate learner frustration.
- Ensure learners are aware of available language options prior to starting the lesson.

Implementation Approach

- [Feature Flag Implementation](#)
- [Common modal Component](#)
- [Topic view Page](#)
- [Lesson and Practice card component](#)
- [Representation of chapter availability](#)
- [Lesson player page](#)
- [Warning Display in Topic Editor](#)
- [Text and Voiceover Language Options](#)
- [Start/Resume/Review Button Terminology](#)
- [Skills Dialog Implementation](#)
- [Responsive Design and URL Redirection](#)

Feature Flag Implementation:

A feature flag with the name **redesigned_topic_viewer_page** will be added, to hide the redesigned topic viewer page from the learner. This flag will be enabled after complete testing. The flag will be added in [feature_flag_list.py](#). See [this bookmark below](#) for how it will be used.

Common Modal Component:

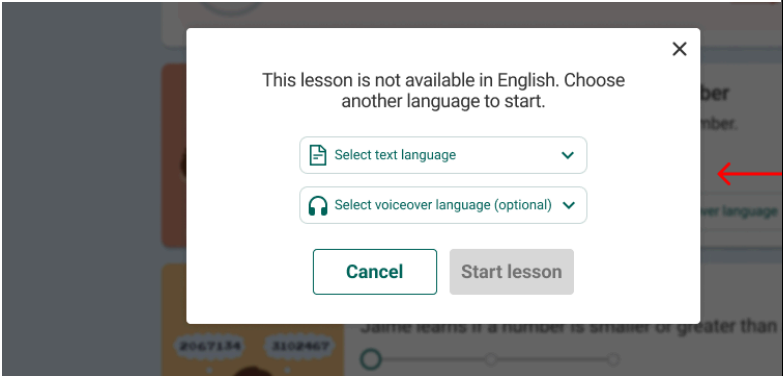
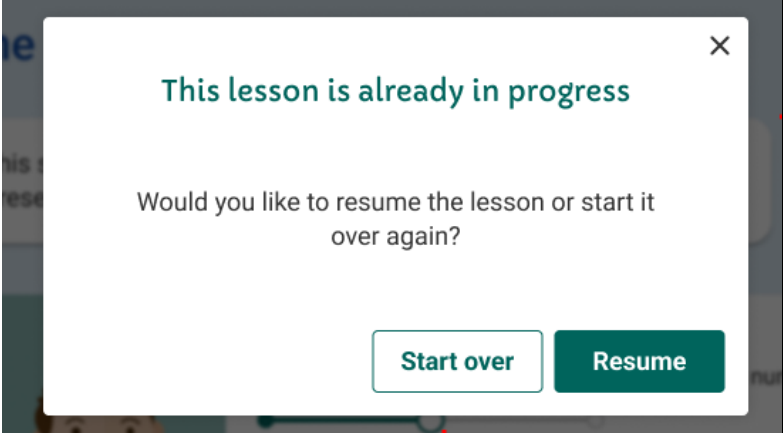
Implementation:

- Create a Service:** A service where we define all the sub-modal components along with their inputs.
- Create Sub-Modal Components:** Create Angular components for each type of modal. These components will be used to display the content of each modal type.
 - Secondary-language modal
 - Start- resume Modal
 - Review Modal
 - Skills of topic
 - Subscribe-modal
 - Review card modal

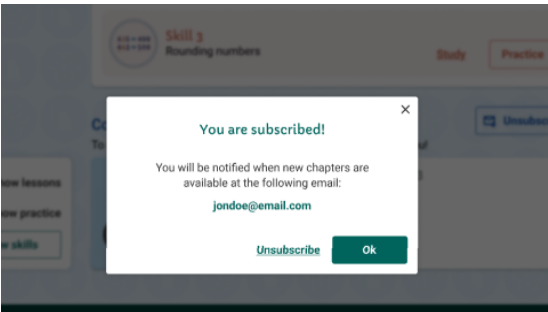
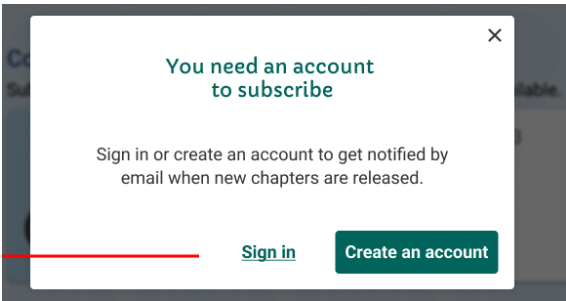
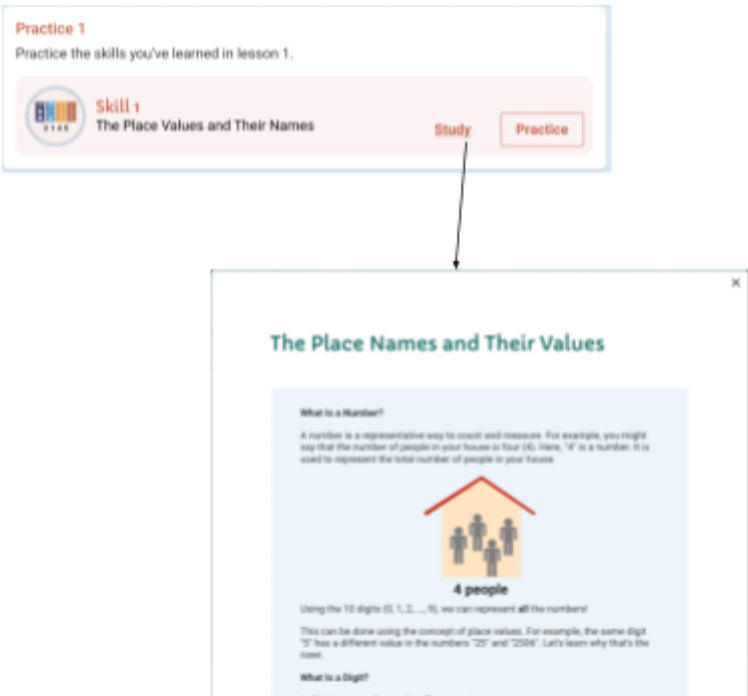
¹²Note: Oppia can only use third-party libraries that are compatible with our Apache 2.0 license. If you're unsure about license compatibility, talk to a platform TL.

3. **Use Modal Service with Dependency Injection:** Inject the ModalService into the Common Modal component. Then, call the appropriate method on the service to open the desired sub-modal.

Based on the Figma mocks, the following modals are required:

Sub-Modal Title	Description	Figma									
<div>secondaryLanguageModal</div> <div>Description: This modal appears when a learner starts a lesson without knowing that it's unavailable in their preferred language. It displays the message: 'This lesson is not available in your preferred language. Please choose another language to start.'</div>	<div>Input: 1. List of available Translation language code and voiceover language accent code are fetch from backend (Implementation)</div> <div>Output: 1. Selected translation language 2. Selected voiceover language'</div> <div>.</div>										
<div>startResumeModal</div> <div>Description: This modal appears when learners click the 'resume' button to complete their unfinished lessons. It pops up a dialog giving the learner the option to either continue their lesson from where they left off or start the lesson again from the beginning.</div>	<div><u>Difference between Review and Resume Button.</u> This '/get_user_saved_checkpoints?exploration_id=<exploration_id>' new URL triggers the new handler 'UesrSavedCheckpoints' to fetch the user checkpoint progress in the StoryNodeTileComponent.</div> <table><tr><td></td><td>Review</td><td>Resume.</td></tr><tr><td>Clause:</td><td>If all the checkpoints are marked then the 'review' button will be visible.</td><td>If only some of the checkpoints are marked then the 'resume' button will be visible.</td></tr><tr><td>Action:</td><td>After clicking the review button the Exploration viewer page will be opened but from the beginning along with saved answers.Implementatio n</td><td>After clicking the resume button the Exploration viewer page will be opened but from that checkpoint where learner leaves the exploration page in their previous session. Similarly like this.</td></tr></table> <div>If Resume 1. Implementation: a. The resume button becomes active when only some of the checkpoints are visited and redirected to the Exploration viewer page along with the lesson progress of logged in users.</div> <div>If Start over 1. Implementation: a. ExplorationRestartEventHandler will be called, which clear all the check points of exploration card(state) for logged in user</div> <div>2. Redirect: a. Navigate to Exploration viewer page</div>		Review	Resume.	Clause:	If all the checkpoints are marked then the 'review' button will be visible.	If only some of the checkpoints are marked then the 'resume' button will be visible.	Action:	After clicking the review button the Exploration viewer page will be opened but from the beginning along with saved answers. Implementatio n	After clicking the resume button the Exploration viewer page will be opened but from that checkpoint where learner leaves the exploration page in their previous session. Similarly like this .	
	Review	Resume.									
Clause:	If all the checkpoints are marked then the 'review' button will be visible.	If only some of the checkpoints are marked then the 'resume' button will be visible.									
Action:	After clicking the review button the Exploration viewer page will be opened but from the beginning along with saved answers. Implementatio n	After clicking the resume button the Exploration viewer page will be opened but from that checkpoint where learner leaves the exploration page in their previous session. Similarly like this .									

<div>review Modal</div> <div>Description: When the learner completed their lesson, this modal popped up. The modal asked the learner, 'Would you like to review your answer or start the lesson over again?'</div>	<div>Event:</div> <div>If start over:</div> <div><div>1. Implementation</div></div> <div>If Review:</div> <div><div>1. The new handler will be created named 'UesrSavedCheckpoints' which is triggered by the URL /get_user_saved_checkpoints?exploration_id=<exploration_id> where exploration_id passed as a query parameter to fetch the user checkpoint progress.</div><div>2. If all the checkpoints are marked then the review button will be visible.</div><div>3. So when the learner clicks the review button 'review_lesson'=True is passed to the url /explore/<exp_id>?suf=<story_url_fragment>&tuf=<topic_url_fragment>&cuf=<classroom_url_fragment>&nid=<node_id>&review_lesson="true" as a query parameter</div><div>4. When the exploration starts for the first time, its checkpoints are clear. In this case, the first card of the exploration is considered the Firststate. However, when the learner visits some exploration cards, the checkpoints will be marked. If they leave the exploration and return later, they should start from their last marked checkpoint. In that case, the current card of exploration from where the learner starts visiting will be considered the currentstate.</div><div>5. According to the recent scenario If all the checkpoints are marked then the currentstate will be at last card of the exploration. But if the exploration page receives the query parameter 'review_lesson' as True, then the exploration viewer page will be started from the beginning. along with the saved answers. In this context the first card of the exploration viewer page will be pointed by currentstate . So whenever the learner clicks the review button then the exploration viewer page will be opened from the beginning along with the saved answer which they completed in their previous sessions. similarly like this.</div></div>
---	--

<p>Subscribe modal</p> <p>Description: The ‘Subscribe’ modal is triggered when the subscribe button is clicked. The modal performs two tasks:</p> <ol style="list-style-type: none">For logged in userFor logged out user <p>For logged-in users the displayed statement will be “You are subscribed”. In the bottom of the modal two more buttons are present “unsubscribe” and “ok”.</p> <p>For logged-out the displayed statement will be “You need an account to subscribe”.In the bottom of the modal two more buttons are present “sign in” and “Create an account”.</p>	<p>Event:</p> <p><u>For logged-in user:</u></p> <ol style="list-style-type: none">ok<ol style="list-style-type: none">subscribeModal.dialogRef.close() : Modal will be close when the ok button is clicked.Unsubscribe<ol style="list-style-type: none">Subscription will be canceled. <p>Implementation</p> <p><u>For logged-out user</u></p> <ol style="list-style-type: none">sign in<ol style="list-style-type: none">When the learner clicks the “Sign-in” button it will redirect to the login page..Create an account<ol style="list-style-type: none">When the learner clicks the “Sign-up” button it will redirect to the sign up page.. <p>Implementation</p>	 
<p>ReviewCardModal Component</p> <p>Description: Learner clicks ‘study this skill’ then this modal will pop up.</p>	<p>Event:</p> <ol style="list-style-type: none">Study this skill <p>Implementation</p>	

Topic Viewer page:

Heading: Name of the topic.

Subheading: Description of the topic so that learners would get an idea of what they are going to learn.

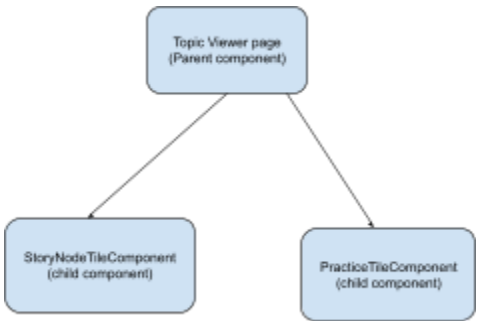
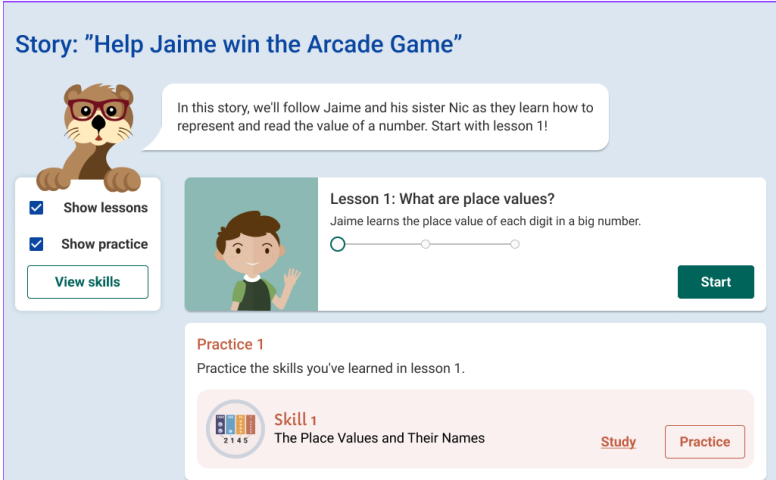
Story-title: Title of the story.

Story description: In the white box, the description of the story will be shown.

UI of Topic-viewer page

Place Values

Did you know that all possible numbers of things can be expressed using just ten digits (0, 1, 2, 3, ..., 9)? In this topic, we'll learn how we can use place values to do that, and, for example, how "5" has a different value in "25" and "2506".



The Topic viewer Page is the parent component, it has the following child components:

- story-lesson-card component
- story-practice-card component

When Learner triggers the URL [TOPIC_VIEWER_URL_PREFIX](#) the topic viewer page will be opened from where the Learner would 'learn' and 'practice' their desired lesson.

The URL `/topic_data_handler/<classroom_url_fragment>/<topic_url_fragment>` trigger the handler `TopicPageDataHandler` to fetch the data regarding topic, so using the **same API** the data of the topic will be fetched. To separate the existing UI and redesign UI ng-If will be used. So in the topic viewer component, [here](#) the feature flag i.e., [enableNewTopicViewerPage](#) will be added , if the [enableNewTopicViewerPage](#) is disabled then the existing UI will be displayed, else the redesign UI will be displayed.

Workflow:

1. Navigate to topic viewer page, where the user can see multiple lesson cards and multiple practice cards
2. Click "**Start**" on the **lesson card**, it triggers the URL & redirects to the lesson player page.
3. Click the "**Start**" button on a **practice card**, it triggers the [URL](#) , guiding the learner to the practice session.
4. Click the "**Study**" button on a practice card, it triggers the [URL](#) , redirecting to the revision page for that specific subtopic.

Implementation:

The topic data will be fetched asynchronously from the backend Api service i.e., [TopicViewerBackendApiService](#)

Inputs with their data structure:

1. `topicUrlFragment`: string = ''
2. `classroomUrlFragment`: string = ''
3. `topicId`: string = ''
4. `topicName`: string = ''
5. `topicDescription`: string = ''
6. `canonicalStorySummaries`: [StorySummary](#)[] = []
7. `Subtopics`: [Subtopic](#)[] = []
8. `chapterCount`: number = 0

```
ngOnInit( ): void {
  this.topicUrlFragment = this.urlService.getTopicUrlFragmentFromLearnerUrl();
  this.classroomUrlFragment = this.urlService.getClassroomUrlFragmentFromLearnerUrl();
  this.topicViewerBackendApiService
    .fetchTopicDataAsync(this.topicUrlFragment, this.classroomUrlFragment)
    .then(
      (readOnlyTopic: ReadonlyTopic) => {
        this.topicId = readOnlyTopic.getTopicId();
        this.topicName = readOnlyTopic.getTopicName();
        this.topicDescription = readOnlyTopic.getTopicDescription();
        this.canonicalStorySummaries = readOnlyTopic.getCanonicalStorySummaries();
        this.subtopics = readOnlyTopic.getSubtopics();
        this.chapterCount = 0;
        for (let idx in this.canonicalStorySummaries) {
          this.chapterCount += this.canonicalStorySummaries[idx].getNodeTitles().length;
        }
      },
    )
}
```

The parent component **Topic-Viewer** page will have two child components:

1. StoryNodeTileComponent
2. PracticeTileComponent

Child component	Input
StoryNodeTileComponent	<div>1. StoryNodeObject</div> <div>2. classroomUrlFragment:It is required to update the Url of exploration page</div> <div>3. topicUrlFragment:It is required to update the Url of exploration page</div>
PracticeTileComponent	<div>1. subtopic</div> <div>2. topicName</div> <div>3. classroomUrlFragment</div> <div>4. topicUrlFragment</div>

Sorting of lessons and practice session in the redesigned topic page

All the data related to the topic will be available in the frontend, so the sorting algorithm calculations will be performed in the frontend.

Algorithm:

nodes = list of all the nodes of a topic (this contains nodes from all the stories)

node_id_to_acquired_skill_ids = {} (this dictionary contains acquired skill ids as values, with the node Id serving as the key for each node)

acquired_skill_ids_from_all_nodes: set<string>=new set() (this set stored the acquired skill ids of all nodes)

for node in nodes:

1. acquired_skill_ids = get acquired skill IDs. (this list store acquired skill ids of each node)
2. node_id_to_acquired_skill_ids[node_id] = acquired_skill_ids
3. acquired_skill_ids_from_all_nodes.add(acquired_skill_ids)

final_subtopic_to_acquired_skill_ids = {}

(This dictionary contains skill IDs as values, with the subtopic IDs serving as the keys. It omits subtopics whose skill IDs are not present in any of the nodes.)

for subtopic in subtopics:

1. acquired_skill_ids = get skill IDs from subtopic.
2. final_subtopic_to_acquired_skill_ids[subtopic.id] = acquired_skill_ids

for subtopic in subtopics:

1. skill_ids = Get skill IDs from the subtopic (this list contains skill ids of each subtopic)
2. skill_ids_included_in_at_least_one_node = [] (This list contains the skill IDs that are assigned in at least one of the nodes.)
3. for skill_id in skill_ids:

a. If skill_id in acquired_skill_ids_from_all_nodes:

i. skill_ids_included_in_at_least_one_node.append(skill_id)
4. if skill_ids_included_in_at_least_one_node is empty:

a. Delete current subtopic from final_subtopic_to_acquired_skill_ids
5. final_subtopic_to_acquired_skill_ids[subtopic.id] = skill_ids_included_in_at_least_one_node

sorted_node_practice_list = [] (this list stores the node and practice object in a sorted order)

for node in nodes:

1. skill_ids = Compute skill_ids from node
2. for skill_id in skill_ids:

a. For subtopic_id, aquired_skill_ids in final_subtopic_to_acquired_skills:

i. If skill_id in acquired_skill_ids:

1. acquired_skill_ids.pop(skill_id) – Removing a skill_id from the acquired skill IDs list once the skill has been learned from a node. Eventually, when the acquired skill IDs list of any subtopic becomes empty this means that all the skills have been learned and the student is ready to give a practice session corresponding to those subtopic which has empty skill_ids.

2. final_subtopic_to_acquired_skills[subtopic.id] = aquired_skill_ids
3. sorted_node_practice_list.push(node)
4. for subtopic in subtopics

a. if final_subtopic_to_acquired_skills[subtopic.id] is empty

i. sorted_node_practice_list.push(subtopic)

StoryNodeTileComponent and PracticeTileComponent

i)StoryNodeTileComponent

A component that stores all the necessary information regarding story nodes (chapter).

Implementation:

1. Fetch the single story i.e., storySummary from canonicalStorySummaries

Input:

```
for storysummary in canonicalStorySummaries
  <StoryNodeTileComponent [storySummary]="storySummary"
                                [classroomUrlFragment]="classroomUrlFragment"
                                [topicUrlFragment]="topicUrlFragment">

  </StoryNodeTileComponent>
```

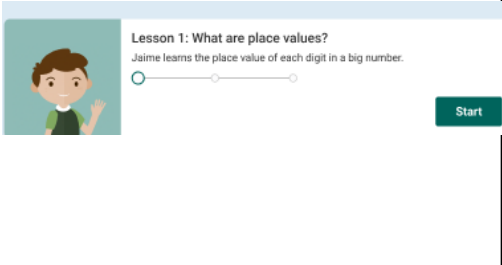
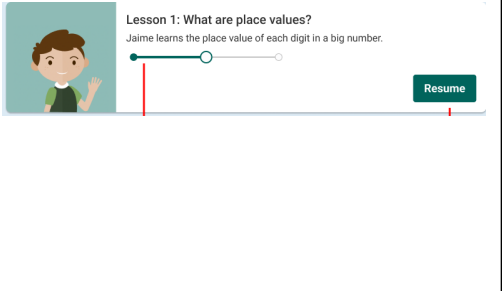
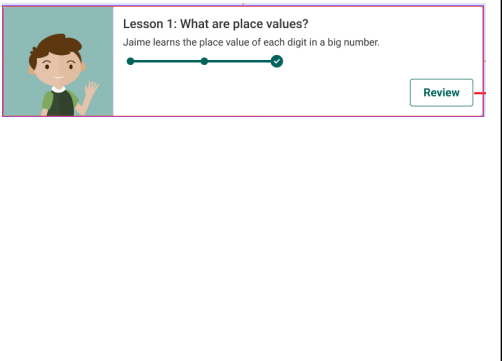

Significance:

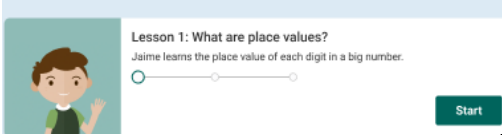


StoryNodeObject : To fetch the content of the associated lesson.

classroomUrlFragment: It is required to update the [Url](#) of exploration page

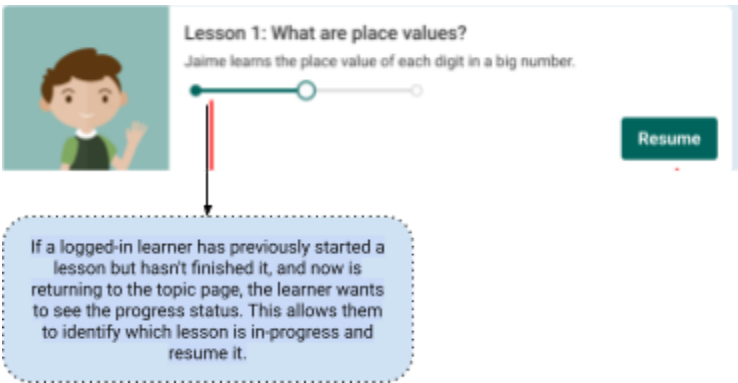
TopicUrlFragment: It is required to update the [Url](#) of exploration page

Number of Buttons required for the lesson card In the lesson card

Buttons	Description	Action	Figma
Start	This button is responsible to start the given lesson and redirect to the selected lesson page	Implementation: 1. ExplorationRestartEventHandler will be called, which clear all the check points of exploration card(state) Redirect: <ul style="list-style-type: none">Navigate to Exploration viewer page	
Resume	The 'Resume' button appears when a logged-in learner identifies an incomplete lesson on the topic page and wishes to continue from where they left off in order to complete it.	Implementation: 1. Redirect to the Exploration viewer page along with the lesson progress of logged in users.	
Review	The review button appears once all the checkpoints of the lesson are completed. This button is responsible for popping up a modal. The modal asks the learner, "Would you like to review your answer or start the lesson over again?"	When the learner clicks the review button the Review-modal will be opened.	
Info	The info button is displayed if the learner starts a lesson without selecting the secondary language (as the content of the exploration viewer page is not available in their preferred language). When the 'info' icon is clicked, it opens a modal informing the learner that the selected lesson is not available in their preferred language. It prompts them to select another language for translation from ' select text language '	<div>1. To open the Tooltip: a. When user click info button</div> <div>2. To close the tooltip: a. Learner click cross button b. Learner click anywhere in the windows i. For this we need to use 'Backdrop:True'</div> <div>Created a class named 'infotooltip'. Another subclass will be created within this 'infotooltip', i.e., 'infotooltiptext', which displays the content of the tooltip. Both classes use strong CSS like</div> <pre>.infotooltip { position: relative; display: inline-block; cursor: pointer; } .infotooltip .infotooltiptext { visibility: hidden; /* Add other CSS properties here */ } .infotooltip:hover .infotooltiptext { visibility: visible; opacity: 1; }</pre> <div>Implementation: <ul style="list-style-type: none">A new component will be created for tooltip i.e., "LanguageSelectorInfocomponent".If the learner interacts with any part of the page while the tooltip is displayed, the tooltip will close automatically.It needs to check if the user's preferred language is not in the list of languages which is present in the language selector drop down. If that is the case, then this tooltip will be displayed.</div>	

Buttons	Description	Action	Figma
Start	This button is responsible to start the given lesson and redirect to the selected lesson page	Implementation: 1. ExplorationRestartEventHandler will be called, which clear all the check points of exploration card(state) Redirect: <ul style="list-style-type: none">Navigate to Exploration viewer page	
Select text language	The 'Select text language' button appears when a learner identifies that a lesson is unavailable in their preferred language but is available in another language. This allows the learner to choose an alternative language for the lesson, enabling them to begin learning in a language they understand. Note: Tooltip should be opened in selected language – this should be internationalized in en.json and qqj.json	Secondary-language-modal	
Select voiceover language	The 'Select voiceover language' button appears alongside the 'Select text language' button. The difference between these two tabs is that 'Select voiceover language' is responsible for choosing the voiceover, while 'Select text language' is responsible for choosing the written text of the lesson. By default, 'Select voiceover language' is set to English.	Secondary-language-modal	

Checkpoints will be present at the bottom of every lesson so that the learner can identify an incomplete lesson on the topic page and continue from where they left off in order to complete it.



The URL `/explorehandler/checkpoint_reached/<exploration_id>` triggers the handler name [CheckpointReachedEventHandler](#) which stores user progress, from where they left the lesson, and stores the last visited checkpoint. The data for checkpoints is fetched in the frontend using the `editable-exploration-backend-api` service. It will display within the `StoryNodeTileComponent`.

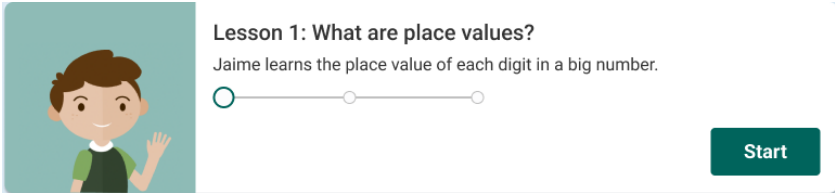
The implementation of checkpoints will be similar to the existing [progress](#) implementation.

Thumbnail

Thumbnail of the lesson will be displayed with their lesson card. After clicking the thumbnail of the lesson it will redirect to the lesson player page.

Title and Description

The titles and descriptions of lessons are essential for learners as they provide clarity on what they are learning to acquire skills. This information ensures that learners are aware of the content being taught and enables them to structure practice sessions based on completed lessons.



Title: What are place values?
Description: Jaime learns the place value of each digit in a big number.

ii)**PracticeTileComponent**

A component which stores all the necessary information regarding practice.

Learners who have completed one or more lessons previously and are now returning to the platform for another study session want to practice the previously learned skills so that they can remember what they have learned.

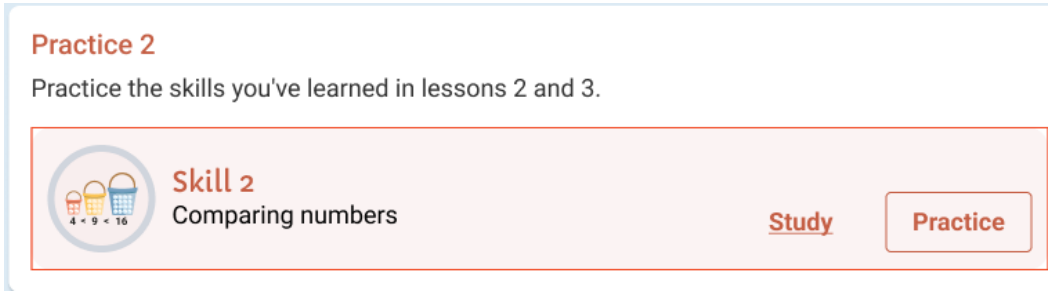
Title

Title : 'Practice with serial number'

Example: Practice 1, Practice 2

Description:

The description of the practice card provides a statement related to the description of lessons, ensuring that learners never get confused about which practice card they are going to perform based on the lessons they have learned



Title: Practice 2.

Description: Practice the skills you’ve learned in lessons 2 and 3.

Implementation:

Input:

```
<PracticeTileComponent      [subtopic]="subtopic"
                             [topicName]="topicName"
                             [topic_id] = “topic_id”
                             [classroomUrlFragment]="classroomUrlFragment"
                             [topicUrlFragment]="topicUrlFragment">

</PracticeTileComponent>
```

Load the practice questions session populated with the desired subtopic IDs:

```
let practiceSessionsUrl = this.urlInterpolationService.interpolateUrl(
  PracticeSessionPageConstants.PRACTICE_SESSIONS_URL,
  {
    topic_url_fragment: this.topicUrlFragment,
    classroom_url_fragment: this.classroomUrlFragment,
    stringified_subtopic_ids: JSON.stringify(subtopicids),
  }
);
```


Skill

Title: In the [practice-card](#) skill 2 is the title of skill

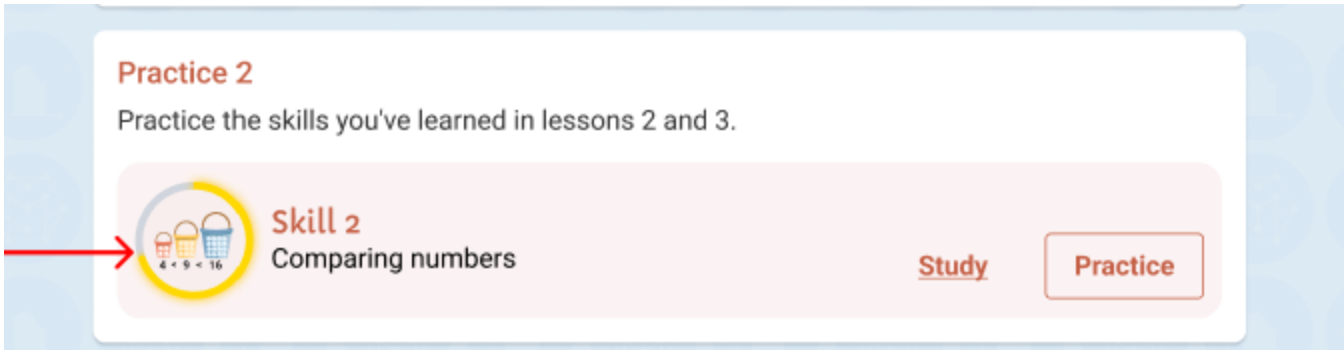
Description: In the [practice-card](#) Comparing numbers is the description of skill card

Buttons:

Button	Description	Implementation:	Mocks
Practice	<p>Learner clicks Practice button: This describes the action taken by the learner, which is to click on the "Practice" button.</p> <p>learner redirected to the practice-session: This describes the result or outcome of the learner's action. After clicking the "Practice" button, the learner is directed or redirected to the practice session.</p>	<p>Input:</p> <ol style="list-style-type: none">subtopicId: It will be used for opening of practice-session-pagetopicName: topicName is the string property , which gives as a input to fetch translatedtopicName and, in practice-tab.component.ts topicName is send as parameter to siteAnalyticServiceclassroomUrlFragment and topicUrlFragment will be used to create practice session URL from which practice session will be opened. <p>Output:</p> <ol style="list-style-type: none">Navigate to the practice-session page	The mockup shows two parts. The top part is a screenshot of the 'Practice 2' card with the 'Practice' button highlighted. A callout box labeled 'After clicking' points to the button. The bottom part is a screenshot of the 'practice-session' page that appears after the button is clicked, showing a list of practice questions.

study	<p>Learner clicks Study Button: This describes the action taken by the learner, which is to click on the "Study" button.</p> <p>learner is redirected to the review-card: This describes the result or consequence of the learner's action. After clicking the "Study" button, the learner is directed or redirected to the review card.</p>	<p>After clicking the Study button a modal named ReviewCardModalComponent will be opened with the following data as input subtopicId. The modal does not perform any actions; only a button to close the modal will be present in the modal footer.</p> <p>Input:</p> <ol style="list-style-type: none">1. subtopic: Retrieve all the data from the subtopic object needed to display on the review card modal <p>Implementation: If the learner clicks the study button, the URL '/subtopic_data_handler/<classroom_url_fragment>/<topic_url_fragment>/revision/<subtopic_url_fragment>' triggers the handler named SubtopicPageDataHandler to fetch the content. In the Subtopic-viewer-backend-api service this fetchSubtopicDataAsync method retrieves the data regarding the subtopic content from the backend. Using this fetchSubtopicDataAsync method we get the content page of the subtopic, similarly like this. This implementation will be present in PracticeTileComponent.</p>	 <p>The diagram illustrates the process of opening a review card modal. It starts with a 'Practice 1' card titled 'Practice the skills you've learned in lesson 1.' which includes a 'Skill 1' section 'The Place Names and Their Values' and 'Study' and 'Practice' buttons. An arrow labeled 'After clicking' points from the 'Study' button to a modal window titled 'The Place Names and Their Values'. The modal contains a 'Message' box with text about place names and a 'Close' button.</p>
-------	--	---	---

Subtopic Mastery:
This section is taken from [PRD](#).



The progress of the learner will be demonstrated within the PracticeTileComponent, based on the progress of individual subtopics.

- Implementation:
- The subtopic mastery will be fetched from [learnerDashboardBackendApiService.fetchSubtopicMastery\(topicIds\)](#).
 - Service calls a URL "/subtopic_mastery_handler/data".
 - Accessing this URL triggers the handler SubtopicMasteryDataHandler.
 - The response from SubtopicMasteryDataHandler is a dictionary of subtopic mastery.

Representation of Chapter Availability
Available vs. Coming Soon lesson

1. Functionality:
 - a. This functionality is only visible when the feature flag isSerialChapterFeatureFlagEnabled is set to True. It segregates the number of chapters listed into two sections: 'Available' and 'Coming Soon'. In available sections chapters are published so they are clickable but in the 'Coming Soon' section chapters are 'Ready to publish' so they are grayed-out, unclickable, and with a wrench icon. The status of individual nodes is retrieved from the storyNode model using the function [getStatus\(\)](#).
2. Implementation:
 - a. The loading of the topic viewer page fetches all the data related to the topic using the URL '/topic_data_handler/<classroom_url_fragment>/<topic_url_fragment>', which triggers the handler TopicPageDataHandler.
 - b. The service responsible for fetching the data from the backend is the topic-viewer-backend-api service.
 - c. Using ng-if node.getStatus()=== 'published' all the publish lesson will be included in this section
 - d. If node.getStatus() === 'Ready to publish' using ng-if, then all the nodes that are ready to publish but have not yet been published will appear in this section

- Chapter Count Field
1. Functionality:
 - a. Display counts of available and coming soon chapters next to their respective section headers.
 2. Implementation:
 - a. Calculate and display counts based on the number of chapters in each section.

Chapter Publication Event

- 1. Functionality:
 - a. Allow learners to subscribe to email notifications for new chapter launches.
- 2. Implementation:
 - a. Implement a **prompt** for signed-in learners to confirm subscription with their email address.
 - b. Provide options for signed-in and not signed-in learners to subscribe or not.
 - c. Implement a mechanism to change the 'Subscribe' button to 'Unsubscribe' after subscription.

Backend tests will be added to ensure whether mail should be sent to learners or not.

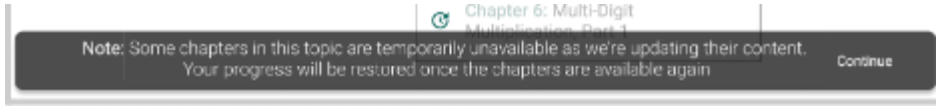
Cases when mail triggered to subscribed learners:

- A new chapter is published to the existing story
- If a chapter is published, then unpublished and again published, then mail should be triggered for each time it gets published.

Cases when mail not triggered:

- An existing chapter is unpublished.
- Story is updated with other data like Url fragments ,Title, description etc. that doesn't involve publishing a new chapter.

Unpublished Chapters



- 1. **Description:** If the learner completed some chapter and then due to some reason chapter were unpublished so leaner should notified that “some chapters in this topic are temporarily unavailable as we’re updating their content”
- 2. Functionality:
 - a. Inform learners when chapters are unpublished and adjust their progression score.
- 3. Implementation:
 - 1. notification_for_unpublished_completed_chapter = false (boolean datatype responsible for display notification message if any completed chapter is unpublished to learner)
 - 2. list_of_completed_node_ids = [] (//store all the chapters completed by learner)
 - 3. progress_model = user_models.[StoryProgressModel](#).get(user_id, story_id, strict=False)
 - a. list_of_completed_node_ids = progress_model.completed_node_ids
 - 4. published_node_ids = [] (//store completed chapters which are published)
 - 5. For node_id in list_of_completed_node_ids:
 - a. Get status of node_id ,if it is published then,
 - i. publihed_node_id.append(node_id)
 - 6. If both published_node_ids and list_of_completed_node_ids are not equal
 - a. notification_for_unpublished_completed_chapter = true
 - b. Get the unpublished and completed node_ids , unpublished_completed_node_ids=[node_id for node_id in **completed_node_ids** if node_id not in **published_node_ids**] (//unpublished_completed_node_ids store non published node ids)
 - c. Create a method which deletes unpublished_completed_node_ids from the user model, similarly like adding node_id to user model ([ref](#)).

Lesson player page

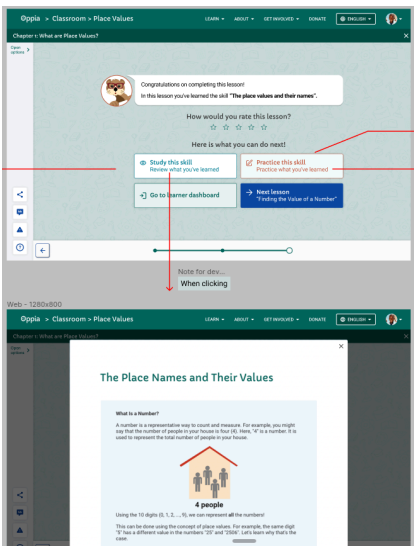
Current workflow:

In the current workflow, the learner can only navigate to another lesson.

Future workflow:

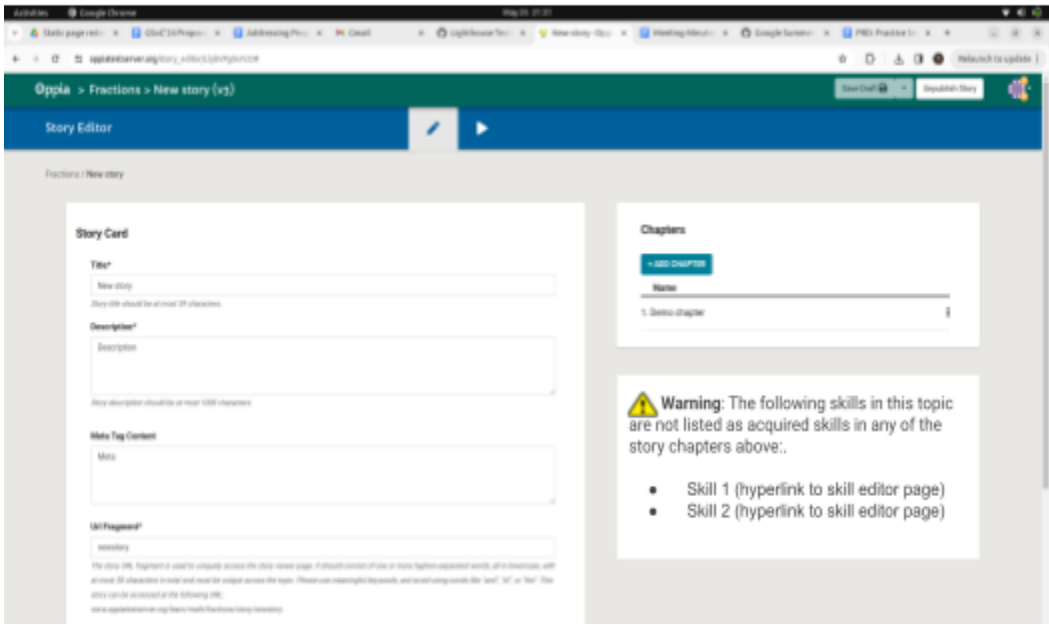
In the future workflow, the learner will be able to navigate to the following section:

- 1. [Study this skill](#)
- 2. [Practice this skill](#)
- 3. Next lesson (currently exist)



Warning Display in Topic Editor:

In the canonical story section of the topic editor page, there should be a warning section displayed, listing all the names of acquired skills that have not been added in a single chapter. This warning does not prevent the publication of the story, but it serves as an important reminder for the topic manager to keep track of the list of unadded acquired skills.



Implementation:

1. skill_ids = [] (//store all the skill ids from every subtopic)
2. for subtopic in subtopics:
 - a. skill_ids_from_each_subtopic = get skill IDs from subtopic. (//store skill ids which is present in each subtopic)
 - b. skill_ids.append(skill_ids_from_each_subtopic)
3. skill_ids_not_assigned_to_chapters = [] (//Store all the skill IDs that are not assigned to any of the chapters)
4. skill_ids_assigned_to_chapters= {} (Store all the skill IDs that are assigned to any of the chapters)
5. For node in nodes
 - a. skill_ids_from_each_node= Compute skill_ids from node (//store skill ids which present in each node(chapter))
 - b. skilld_ids_assigned_to_chapters.add(skill_ids)
6. For skill_id in skill_ids
 - a. If skill_id is not in skilld_ids_assigned_to_chapters (//check whether skill id in present in assigned_skill_ids_from_all_chapters , if true then add the skill id to uncategorized_skill_ids)
 - i. skill_ids_not_assigned_to_chapters.append(skill_id)

Text and Voiceover Language Options:

[Select text language](#) & [Select voiceover language](#)

User Flow:

- a. When the lesson is not available in the preferred language open this modal
- b. The learner should select both the translation language and voiceover language accent in order
 - b.i. Selecting in order is necessary because based on the translation language selection, the voiceover language accent dropdown will contain options.
 - b.ii For this once the learner selects the translation language, the voiceover option will be enabled and after selecting the voiceover language accent, the Start lesson button will be enabled.
 - b.iii. This ensures that the learner must select a voiceover before starting the lesson, even though there is only one option in the voiceover language accent list.
 - b.iv. In case when no voiceover language accent is present then a message saying "No language accent available". For this case, voiceover_language_accent_code cannot be added in the URL
- c. Clicking on the button "Start lesson" will navigate the learner to the exploration player page, the language and language accent selected in the previous page will be passed via URI.
- d. Based on those content language and voiceover language accent codes the exploration will load.

Implementation:

ExplorationLanguagesHandler **is** responsible for displaying the translation and voiceover languages for an exploration to the learner.

1. Fetch Exploration model by exp_id
2. Translation_language_codes = []
3. Fetch Entity Translation Models by exp_id, exp_version
4. Iterate on entity translation models:
 - a. Language code = entiy_translation_model.language_code
 - b. Check whether translation in this language meets this [get_displayable_translation_languages](#) criteria.
 - c. Append in Translation_language_codes
5. For language code in Translation_language_codes:
 - a. Fetch entity voiceovers by exp_id, exp_version, language_code
 - i. voiceover_content_ids = Get content ids for which voiceovers can be added()..
 - b. Fetch exploration object by exp_id,exp_version
 - i. Let alll_content_ids = Get all content ids for which voiceovers can be added ([ref](#)).
 - c. Iterate on each entity voiceovers:
 1. Checks whether entity voiceovers in a specific language accent contains > 30 % of voiceovers.
 - a. If so, append voiceover language accent code to 'voiceover_language_accent_codes'
6. If no voiceover is present then no need to pass "voiceover_language_accent_code" as a query parameter.in the URL

A new field 'exploration_preferred_language_code' will be added in the TopicPageDataHandler to fetch the Exploration preference language of the user

The two selected language codes(translation_language_code and voiceover_language_accent_code) will be added to the URL. This is triggered when the learner selects a secondary language from the translation modal. The learner is then redirected to the exploration viewer page, where both language codes are retrieved from the URL and sent to the content-translation-language.service. Subsequently, the translation is handled by the entity translation service, which is called during the loading of the exploration viewer page

Subscribe and Unsubscribe button:

When the logged in learner clicks the subscribe button then a modal will be opened to click "ok" and "unsubscribe".

When a logged out learner clicks subscribe button then a modal will be opened to click “sign in” and ' ‘create an account' ' then after clicking, learner navigates to sign in or sign up page. After sign in and sign up they are redirected to the same page and open the modal.

When learner is logged in:

1. URL: /chapter_subscribe?&story_id=<story_id>&node_id=<node_id>&subscribe_status=<subscribe_status>
2. Takes the following query parameters:
 - a. **Story_id**: String
 - b. **subscribe_status**: String (if learner click subscribe then status will be “subscribe” and if learner click unsubscribe then status will be “unsubscribe” and by default it set to None)
 - c. **user_id** : string (present in request body)
3. UserSubscriptionsModel:
 - a. A new field i.e.,‘story_ids’ will be added in the UserSubscriptionsModel to store the list of story_ids .
4. Triggers a handler named StorySubscribeHandler in the controller layer.
 - a. Depending on subscribe_status two methods will be created in subscritpion_services.py:
 - i. subscribe_to_story
 - When a learner subscribes to any story then story_id will be appended to the list of “story_ids”.
 - ii. unsubscribe_from_story
 - When the learner unsubscribes from any story then the story_id will be removed from the list of “story_ids”.

When learner is logged out:

1. When a learner clicks the signIn/signup button, the URL /login?return_url=%2Flearn/<classroom_url_fragment>/<topic_url_fragment> will be triggered, where 'learn/<classroom_url_fragment>/<topic_url_fragment>' is passed as a query parameter, and then the learner is redirected to the login page.
2. On the login page, asynchronous tasks are performed, such as retrieving user information from the backend using the function [getUserInfoAsync\(\)](#), which is defined in the user-backend-api service.
3. The URL '/userinfohandler' triggers the handler UserInfoHandler for fetching user information.
4. After successfully retrieving the user information, the login page checks the [authentication](#) and check whether the session is begin or not using [beginSessionAsync](#), if session begin then authentication is true otherwise false, if authentication is true it will redirects to the signup page, where it extracts the return URL from the query parameters using URLSearchParams and includes it in the redirect URL (/signup?return_url=\${returnUrl}), where returnUrl is the query parameter. And if authentication is false then it catches an error.
5. Then, the /signup?return_url=\${returnUrl} URL is stored in a destination variable, and this destination variable is passed as a parameter to windowRef.nativeWindow.location.assign(), which redirects to the topic-viewer page.
6. To open the subscribe modal again, a boolean variable named reopenSubscriptionModal should be stored in the new service file. In the **ngOnInit** of StoryNodeTileComponent, check whether ‘reopenSubscriptionModal’ is true or false. If true, then the modal will be opened; otherwise, the modal will not open. When the learner clicks the subscribe button, ‘reopenSubscriptionModal’ is set to true. If the learner is not a logged-in user, steps 1 to 5 will be implemented, and the user will be redirected to the topic viewer page. While loading the page, the ‘reopenSubscriptionModal’ value is now true, and the modal opens without clicking the subscribe button.

Start/Resume/Review Button Terminology:

[Start button](#)
[Resume button](#)
[Review button](#)

Skills Dialog Implementation:

[viewSkillComponent](#)

Responsive Design and URL Redirection:

Responsive for tablet and mobile:

Html:

1. <meta name="viewport" content="width=device-width,initial-scale=1,user-scalable=yes">

Css:

1. @media screen

Implementation:

Will make a function which checks whether the width of window dimension is for mobile, tablet or desktop using a service called windowDimensionService.

For mobile the width of window should be less than 500

For tab the width of window should be less than 700

Example:

```
@media (max-width: 768px) {
  //CSS for Tablet view
}
@media screen and (max-width: 470px) {
  //CSS for Mobile view
}
```

Metrics plan

Event (see PRD)	Event parameters (see PRD)	Do we already record the event + parameters? <ul style="list-style-type: none">• If so, please link to the corresponding code on GitHub.• If not, describe the changes needed to do so.
Learner clicks to start a lesson.	exploration_id	[Existing]: registerStartExploration (exp_id)

Learner doesn't move to the second card and clicks on "X" to exit a lesson.	<p>explorationId: To check on which exploration the number of requested translation language is not present. So that's why learners exit the lesson after visiting the first card.</p> <p>requested_translation_language_code</p> <ul style="list-style-type: none">The language which is requested by the user.	[New]: Register the event i.e., registerExitLessonEvent() It will be defined in the exploration player page.
Learner finishes a lesson/set of lessons that fully introduced a skill	subtopicId topicId	[New]: Register the event i.e., registerSubtopicSkillsAcquiredEvent() <ul style="list-style-type: none">It will be defined in the exploration player page.
Learner begins a practice session in the introduced skill	classroomUrlFragment topicName subtopicId (Skill is present in the subtopic)	[Existing] : registerPracticeSessionStartEvent() <ul style="list-style-type: none">Params needed for this event<ul style="list-style-type: none">subtopicIdclassroomUrlFragmenttopicName
Learner returns to the platform and starts the second practice session.	classroomUrlFragment topicName subtopicId	[Existing] : registerPracticeSessionStartEvent() <ul style="list-style-type: none">Params needed for this event<ul style="list-style-type: none">subtopicIdclassroomUrlFragmenttopicName
Learner returns to the platform and starts the third practice session.	classroomUrlFragment topicName subtopicId	
Learner returns to the platform and starts the fourth practice session.	classroomUrlFragment topicName subtopicId	
Learner returns to the platform and starts the fifth or more practice session.	classroomUrlFragment topicName subtopicId	

Testing Plan

Acceptance tests for core user(learner) flows

Filename: topic-viewer-user-flows.spec.ts

Initial setup steps/ beforeAll

- [Create a new user](#) and assign the role("curriculum admin, release coordinator") with email : curriculum_release_user@example.com
- Navigate to release coordinator page and enable the feature flag "**enableNewTopicViewerPage**"
- Create exploration with title "**exploration1**" from creator dashboard
 - Navigate to the creator dashboard page.
 - Click on "**Create Exploration**" button
 - Navigate to exploration editor page
 - Create 5 states (cards) are as follows
 - First state
 - Mark this status checkpoint
 - Continue Interaction
 - Second state
 - Mark this status checkpoint
 - Text Input Interaction
 - Feedback
 - Hint
 - Solution
 - Third state
 - Mark this status checkpoint
 - Continue Interaction
 - Fourth state
 - Mark this status checkpoint
 - Text input Interaction
 - Feedback
 - Hint
 - Solution
 - Fifth state
 - End exploration Interaction

5. Publish the exploration.

4.Navigate to topic and skill dashboard.

5. Create topic “**topic1**” from topic and skill dashboard page.

6.Create skill “**skill1**” from topic and skill dashboard page

7.Navigate to the skill editor page and create 3 questions.

8.Navigate to the topic editor page and create subtopic “**subtopic1**”, then add “**skill1**” to the created subtopic ie., “subtopic1”. Then publish the “topic1”

9.Add story “story1” within “topic1”

10.Navigate to the story editor page of “topic1” and publish “story1” .

11. Add chapter “chapter1”, “chapter2” and “chapter3” to the “story1”

12. Add the published exploration Id into the chapter1 and publish it.

13.Publish the “chapter1”, “chapter3” and make “chapter2” as “Ready to publish”

14. Navigate to exploration editor page and switch to translation tab
- a. Select translation languages “English”,“hindi”from the drop down.
 - b. After selecting, add translation for the selected language.
 - c. Again switch to voiceover mode.
 - d. Select voiceover languages and language accent code for “Hindi(India)” from the drop down.
 - e. After selecting, add selected voiceover for the exploration1.
 - f. Publish the changes.

15.User with email:curriculum_release_user@gmail.com logged out.

#	Test name	Initial setup steps/ beforeAll	Steps	Expectations
1.	Logged out learner should be able to study lesson	Ensure beforeAll() is already run.	User clicks the Learn tab from the top navbar of the mainpage of oppia website.	Drop down must be opened for the classroom selection
			Click ‘Basic Mathematics’	learn/math must be opened
			Click topic1 from the math classroom	Display the redesigned topic viewer page
			Click “View skills” button	“View skill” modal must be opened.
			Visit the first lesson i.e, “lesson1” from the topic viewer page.Click start button.	Exploration viewer page must be opened from the beginning “First state”
			Navigate to topic viewer page by clicking “back button”	Topic viewer page must be opened.
2.	Logged out learner should be able to practice	Ensure beforeAll() is already run.	Click practice cards from the topic viewer page. Then click the practice button.	Practice session page must be opened
			Navigate to topic viewer page by clicking “Back button”	Topic viewer page must be opened.
			Click study button	Revision modal of “subtopic1”must be opened.
3.	Logged in learner should be able to resume and review a lesson	Ensure beforeAll() is already run.	Log in : “ curriculum_release_user@example.com ”	Learner must be log in
			Again navigate to preference page using Url	User preference page must be opened
			Change the preferred language from user preference i.e.,Bangla	Site language must be visible in “Bangla”
			Again navigate to the topic viewer page by clicking “Back button”	Topic viewer page must be opened

			click start in “chapter1”	Language selector modal must be opened.
			Select “Hindi” language from translation drop down	Hindi translation language must be selected
			Select “Hindi(India)” accent from voiceover drop down	Hindi(India) voiceover accent must be selected.
			Click the start button from the modal to start the exploration.	Exploration viewer page must be opened
			Start the exploration from the “First state” in the selected languages. Continue up to “Third state”. Checkpoints will be marked up to “Third state” .Navigate to the topic viewer page by clicking “back button”	Topic viewer page must be opened.
			Click the Resume button to open the exploration.	Exploration viewer page must be opened and start from “Third state” and complete “chapter1”.
			Navigate to topic viewer page by clicking “back button”	Topic viewer page must be opened.
			Click the Review button to open the exploration from the “First State”along with the saved progress and answer.	The Exploration Viewer page must be opened.
			Exploration starts from the “First state” along with the save answer.Navigate to topic viewer page by clicking “back button”	The topic viewer page must be opened.
4.	Learner should be able to subscribe and unsubscribe upcoming chapters	Ensure beforeAll() is already run.	Click subscribe button	Subscribe modal should be opened.
			Click sign in button in the modal	Learner should navigate to sign in page
			Enter login email	Should be able to login and navigate to the topic viewer page.
			Click subscribe button	Story must be subscribed.

Implementation Plan

Milestone Table (include both PRs and other actions that need to be taken prior to launch)

Milestone 1

Key objective for this milestone:: Redesign the Topic viewer page which is fully responsive to mobile, tablet,and desktop also. This involves the following steps.

1. Create a feature flag for the redesign topic page. The page will be implemented behind this feature flag until its ready for launch
2. Creation of common modals which we use throughout the oppia.
3. Create 3 components where the topic viewer page will be the parent component and remaining two will be the child component.

a. StoryNodeTileComponent

b. PracticeTileComponent
4. Implementation of “Representation of chapter availability” which is based on the chapter's status whether it is published, unpublished or Ready to publish.
5. A warning message will be displayed in the “canonical stories” section of the topic editor page, if any chapter has missing acquired skills. The warning message will be “Ensure all chapters have acquired skill”.

No.	Description of PR / action	Prereq PR numbers	Target date for PR creation	Target date for PR to be merged
M1.1	Add feature flag to hide the redesigned topic page A feature flag for the redesigned page. The new page should be implemented behind this	NA	28 May 2024	3 Jun 2024

	feature flag until it is ready for launch.			
M1.2	Create a Common modal for new topic page A common modal will be created which contains many submodals which will be created in the milestone 2 like secondary language modal, start/Resume modal, etc.	M1.1	2 Jun 2024	7 Jun 2024
M1.3	Create StoryNodeTileComponent A component that stores and displays all the necessary information regarding story nodes. Display the upcoming chapters which not published, but “Ready to published”	M1.1, M1.2	7 Jun 2024	12 Jun 2024
M1.4	Create story PracticeTileComponent A component which stores and display all the necessary information regarding practice.	M1.1, M1.2	12 Jun 2024	16 Jun 2024
M1.5	Create UI for Topic Viewer Page The new topic viewer page will include the title of the topic as a heading, description of the topic, and the whole story section. It also includes lesson and practice card. <ul style="list-style-type: none"> The New topic viewer page must have a 1.0 lighthouse accessibility score. <p>Note: After every subsequent PR, screenshot of 1.0 score will be added to the description of the PR.</p>	M1.3, M1.4	18 Jun 2024	22 Jun 2024
M1.6	Write acceptance test for the New topic viewer page		20 May 2024	23 May 2024
M1.7	Display notification when chapter unpublished Update the UI of the topic viewer page. If the learner has completed a chapter and the chapter is then unpublished for any reason, the learner should be notified.	NA	22 Jun 2024	25 Jun 2024
M1.8	Display a Warning Message in the canonical story section of the “Topic Editor page” Update the UI of the topic editor's canonical story section. In this section, a warning should be displayed, listing the unadded skills without blocking story publication. This helps the topic manager track them.	NA	24 Jun 2024	27 Jun 2024
	Meeting with PM: Full demo for M1	1, 2, 3, 4, 5, 6, 7	Jun 28, 2024	–
	Fix additional issues raised by PM		30 Jun 2024	

Milestone 2

- Key objective for this milestone:** Add functionality to the topic player page so that it is ready for launch:
- Implementation of Secondary language selection from the [SecondaryLanguagemodal](#). Also include implementation of InfoTooltip
 - Implementation of modal which is included in both StoryNodeTileComponent and PracticeTileComponent.
 - [StartResume modal](#)
 - [Review Modal](#)
 - [ReviewCardModal](#)
 - [Subscribe modal](#)
 - Implement the dialog that lists the skills within each subtopic.
 - [ViewSkills Modal](#)

No.	Description of PR / action	Prereq PR numbers	Target date for PR creation	Target date for PR to be merged
M.2.1	Make exploration player page compatible to load exploration with selected translation language. <ul style="list-style-type: none"> The PR updates the exploration player page so that, by default, the lesson loads in the learner's selected translation language. 		9 Jul 2024	13 Jul 2024

M2.2	<u>Create secondary language modal</u> <ul style="list-style-type: none"> A modal will be displayed when the learner's exploration preferred language is not in the list of available translation languages. A tooltip will also be created which displays a text "This lesson is not available in {{exploration preferred language}}". Extend this acceptance test for the secondary language modal 		5 Jul 2024	10 Jul 2024
M2.4	<u>Create start resume modal</u> <ul style="list-style-type: none"> The PR displays a modal which starts and resumes the lesson according to the marked checkpoints. Extend this acceptance test for the start resume modal 	NA	15 Jul 2024	19 Jul 2024
M2.5	<u>Create review Modal</u> <ul style="list-style-type: none"> The PR displays a modal that reviews the answers given during the completion of the lesson. Extend this acceptance test for the review modal. 	NA	21 Jul 2024	25 Jul 2024
M2.6	<u>Create revision modal</u> <ul style="list-style-type: none"> The PR displays a modal when the learner clicks "study the skill" in the practice component. Extend this acceptance test for the revision modal 	NA	25 Jul 2024	29 Jul 2024
M2.7	<u>Create subscribe modal</u> <ul style="list-style-type: none"> The PR displays a modal for the learner to subscribe to chapters based on their login status. Extend this acceptance test for subscribe modal. 	NA	1 Aug 2024	5 Aug 2024
M2.8	<u>Create view skills modal</u> <ul style="list-style-type: none"> The PR displays a modal to view the list of skills. Extend this acceptance test for the view skill modal. 	NA	6 Aug 2024	9 Aug 2024
	Meeting with PM: Full demo for M1	1, 2, 3, 4, 5, 6, 7	Aug 12, 2024	
M2.9	Fix additional issues raised by PM		15 Aug 2024	
M2.10	Work with the server admin to deploy the code to production. This form should be filled out for feature testing. If no issues are raised, then it will be made public.		17 Aug 2024	
M2.11	Remove feature flag i.e, redesigned_topic_viewer_page		13 Aug 2024	

Launch plan:

M1 tasks and verification in Test Environment

- Estimated release month: July Release (since the deadline for this milestone is 28 June)
- Tasks need to be performed by release coordinator
 - Enable feature flag i.e., **redesigned_topic_viewer_page**
- Verify all the task that are created in M1
 - Learner start lesson from the new topic viewer page
 - Learner resume lesson from where they leave the lesson in their previous session.
 - Learners start practicing skills which they acquired from their lessons.
 - Visibility of "Representation of chapter availability": Two section is present one for published lesson and another for
 - Notify Learner when chapter unpublished
 - Display a warning message on the Topic Editor page if any skillIds is not present in any of the chapters.
- Post verification events:
 - Disable feature flag.

M2 tasks and verification in Test Environment

- Tasks need to be performed by release coordinator
 - Enable feature flag i.e., enableTopicViewerPage.
- Verify all the tasks that are created in M2
 - Learner starts a lesson, it is not in the learner's preferred language, then a secondary language modal will be opened.
 - Exploration player page compatibility.
 - Pop up info tooltip to open the secondary language modal.
 - Learner should start and Resume the lesson from where they leave the previous session.
 - Learner should take Revision by the opening of "revision modal"

- f. Learners should be able to subscribe so that they will be notified about the launch of the subscribed chapters.
 - g. Learner should view skills.
- 3. Disable feature flag.

M1 & M2 tasks and verification in Prod Environment

- 1. Tasks need to be performed by release coordinator
 - a. Enable feature flag i.e., enableTopicViewerPage.
- 2. Verify all the steps from point(a to f)
- 3. Verify all the steps from point(a to g)