Google Summer Of Code 2024 - Proposal

Acceptance Tests

Akhilesh Kumar Yadav

Section 1: About You

Goals and Objectives

Why do you want to do a GSoC project with Oppia?	So, I've just started out in college. It's been a while since I've been coding and now I've been diving headfirst into open-source. So far, it's been a fun journey filled with a lot of learning. Over time, I've come to see open-source as an amazing platform to learn real-world development practices that adhere to industry standards, that too, all under the guidance of seasoned professionals! I found it as the best place to apply my learnings, and on top, I truly subscribe to the philosophy of open source. Now, participating in GSoC with Oppia seems like a really good choice for me, for many reasons. The Oppia community is incredibly active, the onboarding process is smooth, and the members are so experienced and highly responsive. Oppia maintains a high-quality codebase with a strong emphasis on code quality, accessibility and maintainability. This aligns perfectly with my intended contributions and learning objectives for GSoC. Their wiki is filled with specific guidance for Oppia and they have dedicated teams. It's all so helpful and I see a world of learning opportunities at Oppia. In addition, I'm really able to appreciate the free, quality education in multiple languages that Oppia provides.
What do you hope to learn/achieve during GSoC?	I believe the project I've chosen, which involves writing acceptance tests, will provide me with the opportunity to learn new tools and concepts. I'm eager to learn and I'm committed to developing things while collaborating on the project. ● Acceptance Tests ○ Puppeteer. ○ Writing non-flaky acceptance and e2e tests. ○ Writing maintainable, accessible and reusable utility functions. ○ Acquiring knowledge in product Quality Assurance. ○ Debugging flakes. ● Github Actions ○ Writing and optimizing Github Actions workflows. ○ Handle workflow edge cases and errors. ● Soft Skills ○ Precise communication. ○ Effective problem solving. ○ Team collaboration.

	○ Effective time management.	
Which Oppia teams have you collaborated with, and what have you done on those teams?	Dev Workflow Team	Although my journey with this team and Oppia has just begun, I've already had the opportunity to actively collaborate with my teammates on some issues. We've engaged in meaningful discussions to address the challenges we and other developers face, and to devise potential solutions. In my capacity, I've extended support to both my teammates and newcomers, assisting them wherever possible. I've also participated in release testing, wherein I navigated through the assigned user journeys to identify issues that could potentially hamper the user experience. Despite being new, I'm committed to contributing positively to the team and the project, and I look forward to continuing to learn and grow in this team.
Contact information	Name: Akhilesh Kumar Yadav Contact Number: (+91) 9695793556 Current Education: Undergoing Bachelor's of Technology in Computer Science. Email/Google Chat ID: akacademic05@gmail.com LinkedIn Profile: www.linkedin.com/in/akhilesh-kr-yadav	
Preferred method of communication	For everyday conversations and addressing any small issues that I may encounter, I'll opt for Email and Google Chat. Besides, to ensure I'm in sync with my mentor and to discuss any grim challenges I face, I'd prefer to use Google Meet or Zoom.	
Which time zone will you primarily be in during the summer?	Indian Standard Time (GMT+05:30)	
(If you are a student) When are your school holidays?	I will be on summer break this year, from June 10th to August 20th.	
What other obligations might you need to work around during the summer?	I will be occupied with my college semester exams from May 15th to June 9th. Post this, I am free and have no commitments for the summer.	
Planned time commitment	From May 27th to June 9th, due to my college exams, I plan to dedicate 4 hours per day, 5 days a week. This will effectively amount to 20 hours per week, totaling 40 hours over the two-week period. Following my exams, during my summer break, I intend to increase my commitment to 30 hours per week, which is 6 hours per day over 5 days a week. Over the course of 8 weeks, until August 11th, this will accumulate to 240 hours.	

	In total, I will be able to contribute approximately 280 hours to the project. Even though the project requires an estimated 175 hours, I am confident that I can successfully complete the project within the planned timeline with the additional hours I have allocated.

Section 2: About Your Project

Project Details

Project title	Acceptance tests (Milestone : 2)
Project size	Medium (~175 hours)
Why did you choose this project?	I chose this project because of my interest in software testing and quality assurance. I see that well-written tests are crucial for maintaining high-quality software, and I am excited about the opportunity to contribute to this aspect of the Oppia project. Additionally, I am drawn to the challenge of writing non-flaky End-to-End tests and diagnosing and resolving issues within these tests. I understand that by simulating real user scenarios and ensuring the system behaves as expected through acceptance tests, we can detect regression and reduce bugs. Running these tests with every commit would help us ensure that new code does not disrupt the existing features. These tests would help us maintain a great user experience and would allow us to move our changes into production with confidence, thereby strengthening our CI/CD pipeline. Plus, it simplifies our task of tracking CUJs and makes it easier to add tests for new CUJs. It will definitely make things simple and efficient. Finally, I am really motivated by the project's goal of enhancing the reliability and user experience of Oppia, and I look forward to contributing to this goal.

Required Skills

WEB	
I can write TS + Angular code with unit tests.[1]	https://github.com/oppia/oppia/pull/20081

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

I have good UI judgment and attention to detail.[2]	https://github.com/oppia/oppia/pull/19736 https://github.com/oppia/oppia/pull/19744 https://github.com/oppia/oppia/pull/19620
I can debug and fix CI failures / flakes.[3]	https://github.com/oppia/oppia/pull/19917 https://github.com/oppia/oppia/pull/19860
I can write or modify e2e or acceptance tests.[4]	https://github.com/oppia/oppia/pull/19917
I can communicate effectively using debugging docs.[5]	*N/A*
I can write or modify Beam jobs.[6]	*N/A*
I have participated in QA testing.[7]	Release - 3.3.7 https://github.com/oppia/oppia/issues/20003 https://github.com/oppia/oppia/issues/19966 https://github.com/oppia/oppia/issues/19961

As of 31th March, 2024, I have made some contributions to the Oppia web repository. This includes 8 merged Pull Requests (PRs), a total of 9 PRs opened, and 7 issues raised. I also have a PR open for Acceptance tests which is a part of this project and I'll get this merged ASAP.

In addition, despite not having any Pull Requests in TypeScript (TS) + Angular in the Oppia repository, I have prior experience of working with both technologies together. I have used TS and Angular to develop some projects, one of which includes a To-do List application named "DoMinder".

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.*

---

[2] To develop this skill: *Tackle a non-backlog responsiveness issue from the LaCE team.*
[3] To develop this skill: *See issues labelled "CI breakage" on GitHub, or look at CI failures in develop (or ❌ signs here) and figure out how to fix them.*
[4] To develop this skill: *Solve part of issue #17712 by covering the CUJs for one or more sets of users.*
[5] To develop this skill: *Tackle an issue that requires creating a debugging doc that you share with the broader team. Fixing CI failures counts.*
[6] To develop this skill: *See this doc for some examples of issues to work on.*
[7] 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.*

Coding period	● 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.	● Yes
In addition to the above: how often, and through which channel(s), do you plan on communicating with your mentor?	I'm planning to keep my mentor updated daily to ensure the project progresses smoothly. I'd prefer to use email for daily updates and Google Chat for resolving minor queries. Additionally, I'd prefer to have at least two mentor meetings per week on Zoom or Google Meet. These will be my preferred primary communication channels.

Section 3: Proposal details

Problem Statement

Target Audience	Developers of all the teams, Testers/QA.
Core User Need	● Developers need to ensure that their code meets the requirements, follows the expected flow, and does not introduce any bugs. ● When developers introduce new features, they need to verify that these features work as expected and do not adversely affect the existing Critical User Journeys (CUJs), and that they align well with them. ● Testers/QA need to manually go through multiple critical user journeys to check if things are working as expected and also need to find bugs that were introduced during a development cycle. We need many testers for this, and they take their time, which delays our CI/CD process. ● Product managers need to ensure that the product being developed aligns with the overall product vision and end-user needs. ● Product Managers need to maintain and ensure the proper functioning of Critical User Journeys (CUJs). The current challenge lies in the organization of the end-to-end tests, which are structured around the individual technical requirements of each page in the application. This approach does not align well with user flows, which often span multiple pages or involve a sequence of interactions with various controls. As a result, it becomes difficult to reason about the coverage of these tests and to understand the intended behavior across an entire user journey. This disjointed structure also

	complicates the process of writing code for new CUJs.
What goals do we want the solution to achieve?	Via writing complete suite of acceptance tests, we aim to achieve the following goals: ● Provide a clear and concise set of criteria that the code must meet. This will enable developers to ensure their code meets requirements, follows the expected flow, and does not introduce bugs. ● Serve as an aid in verifying that new features align well with existing Critical User Journeys (CUJs). ● Streamline the testing process by automating the verification of multiple critical user journeys. This will reduce the time taken during the testing process, thereby accelerating the CI/CD process. ● Provide a safety net for detecting regressions early in software development. Acceptance tests define the expected behavior of a system in a Critical User Journey and verify that the system behaves as expected. When developers make changes to the code, they run these tests to ensure that their changes do not inadvertently cause the system to deviate from its expected behavior. If a test fails, it means that a change has caused a regression, i.e., a feature that was working correctly before is no longer functioning as intended. The earlier this regression is detected, the easier it is to fix, as developers can isolate the problematic change more easily. ● Provide a structured way to track CUJs, making it easier to ensure they are working properly and to write code for new CUJs.

Section 3.1: WHAT

The project aims to incorporate a complete set of acceptance tests for the learner's and admin's user journeys into our application via authoring them and including them in our existing acceptance test workflow. This way the existing acceptance tests will be executed for every Pull Request (PR) that is raised, as part of the Continuous Integration/Continuous Deployment (CI/CD) process. If a PR includes a new acceptance test, then the developer will need to incorporate those tests into the existing workflow file. This will ensure that the newly written test is also executed as part of the CI process, allowing it to be tested in conjunction with the rest of the changes.

This integration will ensure that PRs adhere to requirements and prevent bugs in Critical User Journeys (CUJs). It will also automate the testing of CUJs, thereby accelerating the Continuous Integration/Continuous Deployment (CI/CD) process.

The infrastructure for authoring and executing these acceptance tests is already in place, utilizing Puppeteer and Jasmine. These tests can be run locally from a fork of the local Oppia directory using the following commands:

In a Python setup:

Python

```
python -m scripts.run_acceptance_tests --suite={{suiteName}}
```

In a Docker setup:

```
make run_tests.acceptance suite=SUITE_NAME
```

The CUJs to be covered in this project include are::

- Learners' journeys: These include guest/anonymous users and logged-in users. The surfaces to test include the site's static pages, the exploration player, the learner dashboard, the embedded lesson player, the newsletter email signup, and the "contact us" page.
- Admins' journeys: These include the curriculum admin or topic manager (topic, skill, question creation/editing), contributor dashboard admin, release coordinator (feature flags, flush cache, running a Beam job), and site admin.

The project's scope is limited to integrating acceptance tests for current user journeys. It doesn't include the creation of tests for new user journeys that will be introduced post-April 2024. Nonetheless, I will ensure to maintain code quality and enhance the reusability of utility functions. This way, if there are some other Critical User Journeys (CUJs) that follow similar steps, the same functions can be reused.

Additionally, when a particular suite is written, any existing WebDriverIO tests whose functionality is fully covered by the new acceptance tests will also be removed.

In summary, the integration of acceptance tests aims to enhance the efficiency and effectiveness of the development process, ensure product quality, and align the product with user needs. This approach will provide a faster, smoother, and more error-resistant development and testing experience.

Key User Stories and Tasks that will be implemented

#	**Title**	**User Story Description (role, goal, motivation)** *"As a …, I need …, so that …."*	**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	Acceptance Test Implementation	As a developer, I need to implement new acceptance tests for new features that'll be added and modify existing acceptance tests for features that are updated in those specific user journeys, so that all functionalities are properly tested and work as expected.	1. Understand the requirements of the newly added features.	N/A
			2. Write acceptance tests for the newly added feature.	Refer to this document for a general understanding of how to write acceptance tests.
			3. Update existing tests to reflect any changes in the features.	The existing tests can be found here.

			4. Run the tests to ensure they work as expected.	The command mentioned below can be used to run the acceptance tests on the local machine : python -m scripts.run_acceptance_tests --suite={{suiteName}}
2	Acceptance Test Debugging	As a developer, I need to run the acceptance tests locally or fix issues reported in the CI report of my PR, so that any bugs or problems can be identified and fixed. Additionally, I need to identify and address flaky tests, so that the reliability of our test suite is maintained.	1. Run the acceptance tests or check the CI report on github actions.	
			2. Observe the results.	N/A
			3. Identify any failures or issues.	N/A
			4. Report these issues for further investigation and resolution	If the test fails, observe the stack trace and report the issue here.
3	CUJ Testing Optimization	As a QA coordinator, I need to inspect the top level of the files to identify which CUJs are under automatic testing, so that I can mark these CUJs and optimize the testing process.	1. Open the files containing the acceptance tests.	Here is the list of the CUJs covered under the acceptance test.
			2. Review the tests to identify which CUJs are under automatic testing.	N/A
			3. Mark these CUJs in the testing plan.	N/A
4	CUJ Coverage Inspection	As a product manager, I need to inspect the top level of those files to see what CUJs are covered, so that I can understand the scope of the acceptance tests and ensure they align with the product requirements and user needs.	1. Open the files containing the acceptance tests.	The available acceptance tests can be found here.
			2. Review the tests to identify the CUJs they cover.	
			3. Compare the covered CUJs with the product requirements and user needs.	

Additions/Changes to Web Server Endpoint Contracts

No new additions/changes are needed to be made to the web server endpoints

Calls to Web Server Endpoints

No new calls are needed to web server endpoints.

UI Screens/Components

No new UI screens/Components are required.

Data Handling and Privacy

No new data handling and privacy required.

Other Requirements

No additional requirements.

Section 3.2: HOW

Existing Status Quo

We already have an established infrastructure for running and writing acceptance tests. This infrastructure uses Puppeteer, a Node.js library that provides a high-level API for controlling Chrome or Chromium over the DevTools Protocol. Puppeteer is used for automating browser actions and generating the data needed for the tests. The project also uses Jasmine, a behavior-driven development framework for testing JavaScript/TypeScript code. Jasmine is used for defining the test cases and assertions.

Some of the Critical User Journeys (CUJs) from both milestone 1 and milestone 2 are already covered by the existing acceptance tests in the directory oppia/core/tests/puppeteer-acceptance-tests .

Here's a table of the CUJs that are currently covered:

	User Type	**User Journey Description**
1	Blog Admin	1.1 : User can visit the blog editor page.

		1.2 : User can : - Appoint new blog admin and blog post editors. - Remove existing blog post editors. - Add tag for categorizing the blog posts - Set maximum tags limit on the blog-dashboard
2	Blog Editor	2.1 : Users cannot publish blog posts if the title duplicates an existing one
3	User browsing static pages (logged-in user)	3.1 : Click all buttons in the foundation page.
		3.2 : Click all buttons in the about page.
		3.3 : Click all buttons in thanks for the donation page.
		3.4 : Click all buttons on the navbar.
		3.5 : Click-all-links-in-about-oppia-footer
		3.6 : click-all-links-on-get-started-page
		3.7 : subscribe-to-creator-and-view-all-explorations-by-that-creator
4	Exploration Editor	User can : - Preview the exploration - Can start the questions from the beginning
		Users can : User can do the Basic setting: - Title - Goal - Add a category - Language - Name of the first card - Tags - Advance features - Roles - Voice artist - Permissions - Feedback/Suggestion Email Preferences - Control :- Delete exploration
4	Translation admins	4.1 : Admin should be able to provide Translation rights in a language to a user.
		4.2 : Admin should be able to remove translation rights in a language to a user.
5	Practice questions admins	5.1 : Admin can provide Contribution Rights to submit and review Questions.
6	Voiceover Admin	6.1 : Voiceover admin can add voiceover artists to an exploration, and

		see an error when adding an invalid user.
7	Curriculum Admin	7.1 : As a topic manager, the user should be able to create a Topic, sub-topic, skill, story, chapter and publish it

However, based on issue 17712, some CUJs need to be covered by new acceptance tests. Once these acceptance tests are written, the corresponding webDriverIO e2e tests will be removed.

Solution Overview

Understanding the Project and Infrastructure :

I have thoroughly reviewed the testing infrastructure and the surrounding codebase. We're utilizing Puppeteer's high-level API to control Chromium or Chrome. By default, Puppeteer uses Chromium for both headless and non-headless operations. Although we can configure Puppeteer to use Chrome, it defaults to Chromium. This is advantageous because it doesn't require Chromium to be pre-installed on the host device; Puppeteer automatically installs the most compatible version of Chromium. Additionally, we're using Jasmine as our testing framework, which provides the necessary framework for testing our TypeScript code.

Previously, we were using JavaScript but recently we have migrated to TypeScript and made some architectural changes via this PR made by jnvtnguyen. This PR introduced a significant shift in how user roles are handled in our system. Previously, the system was based on a rigid inheritance structure, wherein it was limiting and complex, especially when a user needed to have multiple roles.

The PR addressed this issue by implementing a composition-based structure. In this new structure, each user has a basic set of capabilities. When we need to assign additional roles to a user, we add new functions to the user class that correspond to these roles. This allows a user to have multiple roles at the same time, which makes our system more flexible and easier to manage.

One of the key functions that play a pivotal role in this change is composeUserWithRoles() in the UserFactory class.

```
private static composeUserWithRoles = function <
  TUser extends BaseUser,
  TRoles extends BaseUser[],
 >(user: TUser, roles: TRoles): TUser & UnionToIntersection<TRoles[number]> {
  for (const role of roles) {
   const userPrototype = Object.getPrototypeOf(user);
   const rolePrototype = Object.getPrototypeOf(role);
   Object.getOwnPropertyNames(rolePrototype).forEach((name: string) => {
    Object.defineProperty(
     userPrototype,
     name,
     Object.getOwnPropertyDescriptor(rolePrototype, name) ||
```

```
        Object.create(null)
      );
    });
  }
  return user as TUser & UnionToIntersection<TRoles[number]>;
};
```

This function takes a user and a list of roles, and “composes” them together. It does this by iterating over each role and adding the properties of the role to the user’s prototype. This effectively gives the user all the abilities of each role.

Another important function we’ve is assignRolesToUser():

```
static assignRolesToUser = async function <
  TUser extends BaseUser,
  TRoles extends (keyof typeof USER_ROLE_MAPPING)[],
>(
  user: TUser,
  roles: TRoles
): Promise<TUser & MultipleRoleIntersection<TRoles>> {
  for (const role of roles) {
    if (superAdminInstance === null) {
      superAdminInstance = await UserFactory.createNewSuperAdmin('superAdm');
    }

    switch (role) {
      case ROLES.BLOG_POST_EDITOR:
        await superAdminInstance.assignUserToRoleFromBlogAdminPage(
          user.username,
          BLOG_RIGHTS.BLOG_POST_EDITOR
        );
        break;
      default:
        await superAdminInstance.assignRoleToUser(user.username, role);
        break;
    }

    await superAdminInstance.expectUserToHaveRole(user.username, role);

    UserFactory.composeUserWithRoles(user, [USER_ROLE_MAPPING[role]()]);
  }

  return user as TUser & MultipleRoleIntersection<typeof roles>;
};
```

- This function assigns roles to a user.
- It first checks if a super admin instance exists. If not, it creates one.

- Depending on the role, it either assigns the user to a specific role (e.g., blog post editor) or directly assigns the role to the user.
- The function ensures that the user has the expected role after assignment.
- Finally, it composes the user with the assigned roles and returns the updated user instance.

Lastly we can create a new user using createNewUser():

```
static createNewUser = async function <
  TRoles extends (keyof typeof USER_ROLE_MAPPING)[] = never[],
 >(
  username: string,
  email: string,
  roles: OptionalRoles<TRoles> = [] as OptionalRoles<TRoles>
 ): Promise<LoggedInUser & MultipleRoleIntersection<TRoles>> {
  let user = UserFactory.composeUserWithRoles(BaseUserFactory(), [
   LoggedInUserFactory(),
  ]);
  await user.openBrowser();
  await user.signUpNewUser(username, email);
  activeUsers.push(user);

  return await UserFactory.assignRolesToUser(user, roles);
 };
```

- This function creates a new user instance.
- It combines the base user functionality with the logged-in user functionality.
- The user is opened in a browser, signed up with the provided username and email, and added to the list of active users.
- If any roles are specified, they are assigned to the user using the assignRolesToUser function.
- The resulting user instance has both the logged-in user capabilities and any assigned roles.

In the puppeteer-testing-utilities, we have:

1) console-reporter.ts: Provides console-based test result reporting.
2) puppeteer-utils.ts: Contains utilities for interacting with Puppeteer.
3) show-message-utils.ts: Handles displaying messages during testing.
4) test-constants.ts: Stores constants used across test files.
5) user-factory.ts: Creates user instances and assigns roles.

After the mentioned files, we have top-level spec files located in the puppeteer-acceptance-tests/spec directory. These spec files outline the steps that a user will encounter as they progress through specific

user journeys. By examining these specs, we gain valuable insights into how the system and its features operate, as well as their intended behavior. Additionally, corresponding utility functions for these tests or statements are available in the puppeteer-acceptance-tests/user-utilities directory.

Planning the Acceptance Tests that needs to be written :

Here's a table of the Critical User Journeys (CUJs) that need to be covered as per the project (Milestone 2):
This includes the Critical User Journeys (CUJs) that are in the existing web QA tests spreadsheet, release testing document as well as some of the other journeys that **a learner and admin may experience**.

- All user journey numbers that start with the same number (such as 1.1, 1.2, 1.1.1 and so on) belong to the same user type.
- If a CUJ does not contain a resource link, it indicates that this journey has not been documented in both the Release Testing Document and the Web QA Test Matrix Spreadsheet.
- Each row makes one test file and each test can have a number of tests(it blocks) in it.
- Green: CUJ already covered.
- Yellow : CUJ is partially covered, i.e., some part is might be left to be covered.
- Orange: existing test needs to updated

Critical User Journeys.

Sr. no.	User Journey	Test file names that'll cover the respective CUJs	Approval status of the corresponding user stories.
Admin CUJs			
1	**Curriculum Admin.**		
1.1	As a curriculum admin, the user should be able to : (CUJ v2 doc 5.1) • Create a topic and skill, and publish both. • Create a new skill with 3 questions in it and assign it to a topic. • Add a subtopic within the topic, and assign the skill to this subtopic • Add the same skill as a diagnostic	create-publish-unpublish-and-delete-topic-and-skill.spec.ts	

	test skill for the topic. • Save the changes • Unpublish and delete the existing topic and skill. • Verify that, after deletion, the respective links for Topic and skill should return a 404 error, indicating successful deletion.		
1.2	As a curriculum admin, user should be able to: (CUJ v2 doc 5.3) • Create a Classroom with a name and URL fragment. • Select the created Classroom, add 'course details', 'topic list intro', and topics. Add prerequisite topics to the topics in the Classroom. • View the graph visualization of the topics and their prerequisite topics. • Verify that the Classroom and its contents are correctly displayed in the learner view. • Delete the created Classroom. • Verify that the deleted Classroom is no longer accessible to the learners.	create-edit-and-delete-classroom.spec.ts	
2	**Topic Manager.**		
2.1	As a topic manager, user should be able to: (CUJ v2 doc 11.1) • Create a subtopic, story (add chapters to it) and publish it. • Unpublish and delete a story and delete subtopics and chapters	create-publish-unpublish-and-delete-subtopic-and-story.spec.ts	
2.2	As a topic manager, user should be able to: (CUJ v2 doc 11.5) • Filter topics by status, classroom, and keyword. • Sort the available topics based on various parameters. • Use the paginator to select the number of topics to show per page.	browse-topics-on-topics-and-skills-dashboard.spec.ts	

	• Open an existing topic.		
2.3	As a topic manager, user should be able to: (CUJ v2 doc 11.5) • Filter skills by status, classroom, and keywords on the topics and skills dashboard. • Sort the available skills based on various parameters. • Use the paginator to select the number of skills to show per page. • Open an existing skill.	filter-sort-and-open-a-skill.spec.ts	
2.4	As a topic manager, user should be able to: (CUJ v2 doc 11.6) • Edit topic name, thumbnail, description, page title fragment for web, and meta tags. • Enable or disable the practice tab to learners • Preview lesson, practice, and revision in the topic preview.	edit-and-preview-a-topic.spec.ts	
2.5	As a topic manager, user should be able to: (CUJ v2 doc 11.7) • Add sub-topic to a topic, assign skills to a sub-topic, and change the assignments and re-publish the topic. • Open an existing sub-topic, modify data and publish it again. Sub-topic should be previewable in the preview tab.	edit-and-preview-a-subtopic.spec.ts	
2.6	As a topic manager, user should be able to: (CUJ v2 doc 11.8 and 1.9) • Open an existing story, update it, and publish it again. It should be previewable in the preview tab. • Add a chapter with exploration Id that contains interaction/interactions that are not supported on mobile devices. • Unable to publish the story as it includes interactions that are not supported on mobile devices. Interactions that are supported on mobile devices are: ○ AlgebraicExpressionInput	edit-and-publish-a-story-only-with-mobile-supported-interactions.spec.ts	

	○ Continue ○ DragAndDropSortInput ○ EndExploration ○ FractionInput ○ ImageClickInput ○ ItemSelectionInput ○ MathEquationInput ○ MultipleChoiceInput ○ NumericExpressionInput ○ NumericInput ○ NumberWithUnits ○ RatioExpressionInput ○ TextInput		
2.7	As a topic manager, user should be able to: (CUJ v2 doc 11.10) ● Modify/update title, description, thumbnail image and chapter outline in the chapter editor. ● Preview the chapter card. ● Add acquired and prerequisite skills. ● Save the changes.	edit-preview-and-save-a-chapter.spec.ts	
2.8	As a topic manager, user should be able to: (CUJ v2 doc 11.11 and 11.13) ● Open the question editor from the question editor tab for a selected topic. ● Select a skill. Upon selection, a list of existing questions related to that skill appears. ● Choose to either: ○ If editing, make updates to all fields such as the state content, interaction, difficulty, linked skills, answer groups, hints, and solution. ○ If adding a new question, fill in all fields such as the state content, interaction, difficulty, linked skills, answer groups, hints, and solution. ○ Delete the question. ● Specifically add images within the state content of a question: ○ Add new images where	create-and-delete-questions-in-topic-editor.spec.ts	

	necessary. ○ If the uploaded image is an SVG or created in the SVG editor, edit/modify the image in the SVG editor. ○ Replace an existing image by discarding the current upload and selecting a new one. ● Associate(link) a skill with the question and disassociate or eliminate some. ● Use the 'Image Region' interaction, highlight a region in the uploaded image and Save the interaction. ● Delete the question.		
2.9	As a topic manager, user should be able to: (CUJ v2 doc 11.12 and 11.13) ● Open the question editor from the question editor tab for a selected skill. ● Select a skill and edit a question that belongs to that skill. (Edit a question that includes a image) ● Associate(link) a skill with the question and disassociate or eliminate some. ● Create questions for a skill using the skill editor page, ensuring to use **all allowed interactions** and that the skills contain misconceptions (both optional and required misconceptions). ● Delete the question. ● In the preview tab, select a question and preview it.	create-and-delete-questions-in-skill-editor.spec.ts	
2.10	As a topic manager, user should be able to: (CUJ v2 doc 11.14) ● Create a new skill and assign and unassign it to a topic. ● Merge any two skills and use filters to select a skill for merging. ● Merge an outside skill with a topic ● Delete a skill.	create-merge-assign-and-delete-skills.spec.ts	
2.11	As a topic manager, user should be able to: (CUJ v2 doc 11.15)	edit-and-republish-a-skill.spec.ts	

	• Edit Skill Description and Concept Card Explanation. • Add and Delete Worked Examples. • Add and Delete Misconceptions. • Manage Prerequisite Skills. • Edit Rubrics. • Publish it again.		
2.12	As a topic manager, user should not be able to: (CUJ v2 doc 11.16) • Edit topics and stories that are not owned (can view the respective editor though). • Create and delete topics and skills. • Add a skill as a diagnostic test skill for a topic. • Should not be able to access the classroom-admin page.	cannot-do-curriculum-admin-actions.spec.ts	
3	**Moderator.**		
3.1	As a Moderator, the user should be able to: (CUJ v2 doc 12.1) • Access the moderator page and ensure that the page data loads correctly. • View recent commits. • In the Recent Commits table, see Timestamp, Exploration, Category, Username, Commit message and whether the commit is community-owned or not. • Open the respective exploration in the exploration editor by clicking the links attached to the corresponding exploration titles. • View recent feedback messages.	view-recent-commits-and-feedback-messages.spec.ts	
3.2	As a Moderator, the user should be able to: (CUJ v2 doc 12.2) • View all the featured activities in the library. • Add a new activity to the featured list in the community library. • Unfeature activities from the library.	edit-featured-activities-list.spec.ts	
4	**Site Admin.**		
4.1	As a admin, the user should be able to: (CUJ v2 doc 6.1 - 6.3)	edit-user-roles.spec.ts	

	• Access all tabs on the admin page without encountering any errors. • Assign and unassign roles to users by their usernames. • Select a specific role, and view the corresponding allocated actions and the users assigned to that role. • Give topic-manager access to a user after creating a topic in the topics and skills dashboard.		
4.2	As a admin, the user should be able to: (CUJ v2 doc 6.4) • (When run in development mode) Load dummy new structures data. Generate exploration, dummy skills with questions, math classroom, and blog posts, and ensure the ability to reload explorations and collections in the activity tab. • (When run in production mode) Not do the above, and verifying that the controls for these operations are not available.	load-dummy-data-in-dev-mode.ts	
4.3	As a admin, the user should be able to: (CUJ v2 doc 6.5) • Edit a platform parameter via adding rules and changing default values, and save those edits to storage.	edit-platform-parameters.spec.ts	
4.4	As a admin, the user should be able to: (CUJ v2 doc 6.6) • Use all the features on the misc tab of the Admin page.	use-misc-tab-features.spec.ts	
5	**Contributor Dashboard Admin**		
5.1	As a Translation Coordinator, the user can: (CUJ v2 doc 7.1) • navigate to the 'contributor-admin-dashboard' tab. • View Text Translators' performance under the 'Translation Submitters' tab and be able to edit their role. • Select different languages for translation. • Review and edit the roles of Text	manage-translators-and-reviewers.spec.ts	

	Reviewers under the 'Translation Reviewers' tab.		
5.2	As a Question coordinator, the user can: (CUJ v2 doc 7.2) ● Navigate to the 'contributor-admin-dashboard' tab. ● View Question Submitters' performance and add new ones under the 'Question Submitters' tab. ● Review and edit the roles of Question Reviewers under the 'Question Reviewers' tab.	manage-question-submitters-and-reviewers.spec.ts	
6	**Release Coordinator Admin**		
6.1	As a Release coordinator, user should be able to: (CUJ v2 doc 8.1) ● Navigate to (/release-coordinator). ● Select a job from the dropdown in the "Beam Jobs" tab and run it ● View and copy the output, once the job has completed. ● Close the modal that appears after clicking the 'View Output' button.	run-a-beam-job-and-copy-the-output.spec.ts	
6.2	As a Release coordinator, user should be able to: (CUJ v2 doc 8.2) ● Navigate to the "Features" tab on the release coordinator page. ● Alter the rollout percentage for logged-in users as needed. ● Enable or disable the 'Force-enable for all users' option as required. ● Save the changes made.	edit-feature-rollout-configuration.spec.ts	
6.3	As a Release coordinator, user should be able to: (CUJ v2 doc 8.3) ● Navigate to the "Misc" tab on the release coordinator page. ● Flush the cache and get the memory cache profile	flush-and-get-profile-of-redis-cache.spec.ts	
6.4	As a Release coordinator, user should be able to: (CUJ v2 doc 8.4) ● Enable the promo bar and enter a	update-promo-bar-message.spec.ts	

	promo bar message • Save the promo bar message.		
7	**Blog Admin**		
7.1	User can visit the blog editor page. (CUJ v2 doc 9.1) 1. Appoint new blog admin and blog post editors. 2. Remove existing blog post editors. 3. Add tag for categorizing the blog posts 4. Set maximum tags limit on the blog-dashboard	assign-roles-to-users-and-change-tag-requirements.spec.ts	
8	**Voiceover Admin**		
8.1	(CUJ v2 doc 10.1) Voiceover admin can add voiceover artists to an exploration, and see an error when adding an invalid user.	add-voiceover-artist-to-an-exploration.spec.ts	
9	**Contributor dashboard admin (for the old UI; this test will be removed after implementation of CUJs 5.1-5.2)**		
9.1	(CUJ v2 doc 7.1) Admin should be able to provide Translation rights in a language to a user.	add-and-remove-translation-rights.spec.ts	
9.2	(CUJ v2 doc 7.1) Admin should be able to add and remove contribution rights.	add-and-remove-contribution-rights.spec.ts	
10	**Email Dashboard**		
10.1	On the email dashboard, user can: (CUJ v2 doc 14) • Put Queries Based On Various Parameters (such as user activity, exploration creation and editing, and collection creation.) and see output with query details (including Query ID, Submitted On, Submitted By, Status, and Number of Qualified Users for the Query.). • Recheck query status to see if its status has changed from 'processing' to 'completed'.	create-queries-for-sending-bulk-emails.spec.ts	
Learner CUJs			
11	**User Browsing Static Pages (logged-out**		

	user)		
11.1	Users can: (CUJ v2 doc 1.1) • Click all the buttons on the contact-us page	click-all-buttons-on-contact-us page.spec.ts	
11.2	Users can: (CUJ v2 doc 1.2) • Click all the buttons on the Creator Guidelines page.	click-all-buttons on-creator-guidelines page.spec.ts	
11.3	Users can: (CUJ v2 doc 1.3) • Click all the buttons on the teach page.	click-all-buttons-on-teach-page.spec.ts	
11.4	Users can: (CUJ v2 doc 1.4) • Click all the buttons on the Volunteer page.	click-all-buttons -on-volunteer-page.spec.ts	
11.5	Users can: (CUJ v2 doc 1.5) • Click all the buttons on the Donate page.	click-all-buttons-on-donate-page.spec.ts	
11.6	Users can: (CUJ v2 doc 1.6) • Click all the buttons on the Terms of Service page.	click-all-buttons-on-tos-page.spec.ts	
11.7	Users can: (CUJ v2 doc 1.7) • Click all the buttons on the Privacy Policy page.	click-all-buttons on-privacy policy-page.spec.ts	
11.8	Users can: (CUJ v2 doc 1.8) • Click all the buttons on the partnership page.	click-all-buttons-on-partnership-page.spec.ts	
11.9	Users can: (CUJ v2 doc 1.9) • click-all-links-on-get-started-page	click-all-links-on-get-started-page .spec.ts	
11.10	Users can: (CUJ v2 doc 1.10) • Click all buttons in thanks for donating page	click-all-buttons-on donation-thanks-page.spec.ts	
11.11	Users can: (CUJ v2 doc 1.11) • Click all buttons in the about page.	click-all-buttons-on-about page.spec.ts	
11.12	Users can: (CUJ v2 doc 1.12) • Click all buttons in the foundation page.	click-all-buttons-on about-foundation-page.spec.ts	
11.13	On the footer, Users can (CUJ v2 doc 1.13) • Click-all-links-in-about-oppia-footer + (some links are not covered in this tests, will cover them)	click-all-links-on-about-oppia-footer.spec	

11.14	User can: (CUJ v2 doc 1.14) • Navigate to the footer of the page where the newsletter subscription field is located. • Enter their email address into the respective field and click on the "Subscribe" button to submit their email address for newsletter subscription. • Upon successful subscription, view a "Thanks for subscribing" modal and within this modal, find and click on the "See our video" button to verify its functionality. • Similarly, within the same modal, find and click on the "Read our blog" button to confirm its functionality.	subscribe-newsletter.spec.ts	
11.15	On the navbar, Users can : (CUJ v2 doc 1.15) • Click all buttons that navigate to static pages	click-all-buttons-on-navbar.spec.ts	
11.16	On the Blog page, user should be able to: (CUJ v2 doc 1.16) • View all blogs, with the most recent ones appearing at the top, along with tags associated with each blog. • Search for blogs based on keywords and tags. • Utilize the paginator to navigate to subsequent pages of blogs.	view-and-search-for-blog-posts.spec.ts	
12	**Logged-out Learner**		
12.1	From the math classroom, Learner can: (CUJ v2 doc 2.1 - 2.5) • Browse through a list of topics, select one to learn, and use a search bar to find other community-created lessons. • On selecting a topic, view the topic page which includes information about the topic, a list of published stories, a list of revision cards, and a section to start a “practice questions” session. • Choose a story within the topic and play through it. • View a subtopic page (revision	select-and-play-topic-from-math-classroom.spec.ts	

	card), including any linked concept card pop-ups. ● Start and participate in a “practice questions” session from the practice tab.		
12.2	● . ●	browse-and-search-for-lessons-in-community-library.spec.ts	
12.3	From the Community library, user can: (CUJ v2 doc 2.9) ● Search ‘collections’ to find all the collections from the search bar. ● Visit the collection player by selecting one of the collections. ● See the “collection story path” diagram, can see info about individual exploration via hovering onto the explorations in the diagram. ● Share the collection. ● Play any exploration in the collection.	search-for-and-start-playing-a-collection.spec.ts	
12.4	While playing a lesson, user can: (CUJ v2 doc 2.10) ● Interact with different interactions and verify that they function correctly, allowing proper interaction. ● view all the different rich-text components properly. ● View Oppia's feedback whenever any interaction is used. ● Navigate to previously played cards if needed. ● Use concept cards and hints wherever provided. ● View all the previous responses to any question. ● Confirm that they cannot answer a previously-answered question. ● If stuck, they can be navigated to some card. ● Upon reaching a checkpoint, a modal appears informing the user about the checkpoint reached.	play-through-lesson-while-getting-feedback-and-hints.spec.ts	
12.5	From the exploration player, User can: (CUJ v2 doc 2.11 and 2.20)	share-and-give-feedback-for-exploration-but-not-report-and-rate-it.	

	● Be interrupted by a confirmation dialog after hitting the back button in the browser. ● Generate attribution,has the ability to copy the attribute in HTML, print and be able to share the exploration on the provided links ● Share the exploration ● Give feedback for both a particular state, and the whole exploration ● Cannot report and rate the exploration.	spec.ts	
12.6	From the last state of the exploration, user can (CUJ v2 doc 2.12) ● Load the next chapter upon clicking the “Next chapter” card on the last state. ● Load the practice session page upon clicking the practice session card on the last state ● Return to the story from the last state	choose-what-to-do-from-the-last-card-of-an-exploration.spec.ts	
12.7	From the exploration player, user can : (CUJ v2 doc 2.13) ● Check current progress from the lesson info modal and be able to see ratings, views, last updated, contributors, tags, as well as share the lesson on available social site links. ● On reloading or coming back to an already-played exploration, user cannot return to the most recent checkpoint visited, and resume the exploration from the previously visited checkpoint, unless they visited the exploration using a progress URL. ● Cannot answer a previously answered state when continuing the exploration from a previously saved state. ● Cannot generate progress URL before first checkpoint is reached.	track-and-resume-exploration-progress-via-url.spec.ts	

	• Can generate progress URL after first checkpoint is reached, by clicking the 'Save Progress' button and:
	  ○ i) See the 'Sign-Up', 'Sign-In', and 'Don't Ask Me Again' buttons.
	  ○ ii) Understand that the progress link is only valid for the next 72 hours.
	  ○ iii) Copy the generated progress URL for future use.
	• Using the progress URL, start an exploration and choose to either restart the exploration or continue from the most recently visited checkpoint when coming to an already played exploration that was saved using the respective progress URL.		

12.8	From the exploration player, User can: (CUJ v2 doc 2.21) • Start the exploration without signing in, making progress in a temporary mode. • Choose to sign in at any point during the exploration. Upon signing in, their progress is automatically saved in permanent mode. This means that even if they leave the exploration and return later, their progress will be maintained. • Continue the exploration from where they left off after signing in. • Choose to clear their progress by clicking the "Restart Exploration" button. This will reset their progress back to the beginning of the exploration, even in permanent mode.	sign-in-and-save-exploration-progress.spec.ts	
12.9	From the exploration player, Learner can: (CUJ v2 doc 2.14) • View and play a lesson entirely in a particular language (using the dropdown on the first card),	play-lesson-in-different-languages-and-listen-to-voiceovers.spec.ts	

	provided the lesson has all its translations ready in that language and doesn't use SetInput. This can be tested using a specific exploration. ● Start listening to the voiceover of the exploration in any language from any state. This means the learner can choose a specific state of the exploration and start the voiceover from that point, rather than having to start from the beginning of the exploration.		
12.10	The user can: (CUJ v2 doc 2.15) ● Change the site language on the navigation bar. ● Check the navbar and profile-dropdown to confirm that they are translated correctly. ● Navigate through pages to confirm that the translation happened correctly. ● Engage with the exploration in the original language in which it was created, regardless of the site language they have switched to.	change-site-language-and-engage-with-original-exploration.spec.ts	
12.11	(Accessibility) The user can: (CUJ v2 doc 2.16) ● Navigate the library page using the 's' shortcut to skip to the main content. ● Navigate to the Get Started page using the 'Ctrl+6' shortcut. ● Navigate to the Community Library page using the 'Ctrl+1' shortcut. ● Remain on the current page when typing 'Ctrl+2'. ● Stay on the current page when typing 'Ctrl+3'. ● Navigate to the About page using the 'Ctrl+4' shortcut. ● Cannot navigate to the Preferences page using the 'Ctrl+5' shortcut.	use-keyboard-shortcuts-to-navigate-and-shift-focus.spec.ts	

	● Check if the focus is correctly moved to the search bar when pressing ‘/’, to the skip link when pressing ‘s’, and to the category bar when pressing ‘c’ on the library page. ● Cannot create a new exploration. ● Use the ‘s’ shortcut in the exploration player to skip to the main content. ● Use keyboard shortcuts 'j' and 'k' correctly to navigate focus between "Back", and "Next" buttons in the exploration player.		
12.12	The user cannot: (CUJ v2 doc 2.17 and 3.14) ● Add an exploration to 'play later' from the (/community-library page) ● Visit the learner dashboard. ● Visit the creator dashboard. ● Visit the collection player. ● Visit moderator page ● Visit the preferences page ● Visit the topics and skill dashboard page	cannot-access-dashboards-which-require-login.spec.ts	
13	**Logged-in Learner**		
13.1	The user should be able to: (CUJ v2 doc 3.1) ● Create an account. ○ Verify if email validation is functioning correctly. ○ Check if the suggestion for the email with super admin access (testAdmin@example.com) appears when clicking on the email input field. ○ Confirm that the link below the input field, which leads to the page containing information about granting admin access, is working as expected.	create-and-delete-account.spec.ts	

	○ Ensure that an error is thrown when a username is not available. ○ Verify that the terms of use can be checked. ○ 'Submit and Start Contributing' button can be clicked to log in the user. ● Delete the account, as follows: ○ Navigate to the preferences page by visiting the /preferences URL. ○ Click on the delete-account button, which will redirect to the /delete-account page. ○ Proceed with the account deletion on the /delete-account page.		
13.2	After logging-in, user should should be able to: (CUJ v2 doc 3.2) ● Land on the learner dashboard. However, if a user has specified a different preferred dashboard in the /preferences, they should be redirected to that selected dashboard instead. ● Navigate to creator-dashboard, contributor dashboard, learner dashboard, profile, topics-and-skill-dashboard and preference pages from the profile-dropdown menu. Some of these may only appear in the profile dropdown menu with additional roles like topics and skills dashboard.	access-dashboards-and-other-pages-from-profile-menu.spec.ts	
13.3	From the community library page, User can: (CUJ v2 doc 3.3) ● Add an exploration to 'play later' from the (/community-library page) ● Remove that exploration from 'play later' from the	save-an-exploration-to-play-later.spec.ts	

	(/learner-dashboard page) ● Search for the exploration in (/community-library) page and play it.		
13.4	From the exploration player, on returning to an already-played exploration or on refreshing, User can: (CUJ v2 doc 3.12 and 3.13) ● Choose to either: ○ Restart the exploration from the beginning, or ○ Continue from the most recently visited checkpoint. ● Complete the exploration. ● Upon revisiting the completed exploration, start it from the beginning.	restart-or-continue-exploration-on-revisit.spec.ts	
13.5	From the exploration player, User can: (CUJ v2 doc 3.4) ● Give feedback to the exploration after playing it, identifying themselves. ● Give anonymous feedback to the exploration after playing it ● Rate the exploration on the last state. ● Give feedback and report at any point during exploration. ● Check the exploration editor page to ensure that the suggestions show up correctly and that anonymity is preserved when they choose to stay anonymous. ● Check feedback updates page, to see the feedback they provided (and responses to it)	give-feedback-rate-and-report-an-exploration.spec.ts.	
13.6	From the learner dashboard, user can: (CUJ v2 doc 3.5) ● Replay a completed or incomplete exploration or collection from the learner dashboard once they have played it from the library ● Learn something new from the 'learn something new' section on the learner dashboard.	manage-goals-progress-and-lessons-from-learner-dashboard.spec.ts	

	• Add and remove goals in the current goals, and see the completed goals in the learner's tab. • See their progress (in percentage corresponding to each skill in the selected topic) in the progress tab • See the respective badge allotted in the progress tab, based on the current progress • Start a practice session for any skill directly from the progress tab. • See the respective lessons in progress, completed and play later sections on the community lessons tab.		
13.7	From the creator dashboard, User can: (CUJ v2 doc 3.6) • See the number of subscribers in the subscriber tab. • Subscribe to a creator • View all explorations authored by that creator.	subscribe-to-creator-and-view-all-explorations-by-that-creator.spec	
13.8	From the preferences, User can: (CUJ v2 doc 3.7) • Change their profile picture, bio, preferred dashboard, subject interests, preferred audio language and email preferences. • View their updated profile picture/avatar, bio and interests on the profile page after editing. • export account (upload thumbnail image, add bio, add subject interest add site language, add audio language) from the (/preferences) page	edit-profile-preferences-and-export-their-account.spec.ts	
13.9	From the preferences, User can: (CUJ v2 doc 3.8) • Change the site language to an RTL (right-to-left) language, like Arabic, and navigate through pages properly. Pages should be fully mirrored from the LTR equivalent (text and layout). The user should be able to interact with dropdowns, pop-ups, modals,	set-language-to-rtl-and-navigate-through-site.spec.ts	

	buttons, etc. on the following pages: ○ Splash page, About page, Landing pages, Library, Learner dashboard, ○ Exploration Player page, Classroom page, Story viewer and Topic viewers.		
13.10	From the profile page, User can: (CUJ v2 doc 3.9) ● Edit avatar ● See created and edited explorations.	edit-avatar-and-view-created-lessons-in-profile-page.spec.ts	
13.11	(Accessibility) The user can: (CUJ v2 doc 3.10) ● Navigate to the Learner Dashboard page using the 'Ctrl+2' shortcut. ● Navigate to the Creator Dashboard page using the 'Ctrl+3' shortcut. ● Navigate to the Preferences page using the 'Ctrl+5' shortcut.	use-keyboard-shortcuts-to-navigate-to-different-pages.spec.ts	
14	**Logged-Out Learner playing Embedded Lesson.**		
14.1	From the embedded exploration player, the user should be able to: (CUJ v2 doc 2.18) ● Play an embedded lesson in another language and change the language by returning to the introduction page. ● Navigate to previously-played cards. ● Learners cannot answer a previously-answered question. ● Use concept cards and hints wherever provided. ● View all the previous responses to any question. ● Restart the exploration upon refreshing the page. ● Receive a confirmation via a toast message upon completing the exploration.	play-through-embedded-lesson.spec.ts	

Directory Structure that includes all the tests.

Orange : test will be updated/modified.
Green : test is covered and no update/modification is needed.

* Indicated numbers following the file names are referring to the CUJs above in the table.

```
oppia/core/tests/
└── puppeteer-acceptance-tests
    ├── specs
    │   ├── blog-admin
    │   │   ├── assign-role-to-users-and-change-tag-properties.spec.ts (7.1)
    │   ├── curriculum-admin
    │   │   ├──create-publish-unpublish-and-delete-topic-and-skill.spec.ts (1.1)
    │   │   ├──create-edit-and-delete-classroom.spec.ts (1.2)
    │   ├──topic-manager
    │   │   ├──create-publish-unpublish-and-delete-subtopic-and-story.spec.ts (2.1)
    │   │   ├──browse-topics-on-topics-and-skills-dashboard.spec.ts (2.2)
    │   │   ├──filter-sort-and-open-a-skill.spec.ts (2.3)
    │   │   ├──edit-and-preview-a-topic.spec.ts (2.4)
    │   │   ├──edit-and-preview-a-subtopic.spec.ts (2.5)
    │   │   ├──edit-and-publish-a-story-only-with-mobile-supported-interactions.spec.ts (2.6)
    │   │   ├──edit-preview-and-save-a-chapter.spec.ts (2.7)
    │   │   ├──create-and-delete-questions-in-topic-editor.spec.ts (2.8)
    │   │   ├──create-and-delete-questions-in-skill-editor.spec.ts (2.9)
    │   │   ├──create-merge-assign-and-delete-skills.spec.ts (2.10)
    │   │   ├──edit-and-republish-a-skill.spec.ts (2.11)
    │   │   ├──cannot-do-curriculum-admin-actions.spec.ts (2.12)
    │   ├──moderator
    │   │   ├──view-recent-commits-and -feedback-messages.spec.ts (3.1)
    │   │   ├──edit-featured-activities-list.ts (3.2)
    │   ├──site-admin
    │   │   ├──edit-user-roles.spec.ts (4.1)
    │   │   ├──load-dummy-data-in-dev-mode.ts (4.2)
    │   │   ├──edit-platform-parameters.spec.ts (4.3)
    │   │   ├──use-misc-tab-features.spec.ts (4.4)
    │   ├──contributor-dashboard-admin
    │   │   ├──manage-translators-and-reviewers.spec.ts (5.1)
    │   │   ├──manage-question-submitters-and-reviewers.spec.ts (5.2)
    │   ├──release-coordinator
    │   │   ├──run-a-beam-job-and-copy-the-output.spec.ts (6.1)
    │   │   ├──edit-feature-rollout-configuration.spec.ts (6.2)
    │   │   ├──flush-and-get-profile-of-redis-cache.spec.ts (6.3)
    │   │   ├──update-promo-bar-message.spec.ts (6.4)
    │   ├──email-dashboard
    │   │   ├──create-queries-for-sending-bulk-emails.spec.ts (10.1)
    │   ├── voiceover-admin
    │   │   ├──add-voiceover-artist-to-an-exploration.spec (8.1)
```

```
│   ├── logged-out-user
│   │   ├──click-all-buttons-on-contact-us-page.spec.ts (11.1)
│   │   ├──click-all-buttons-on-creator-guidelines-page.spec.ts (11.2)
│   │   ├──click-all-buttons-on-teach-page.spec.ts (11.3)
│   │   ├──click-all-buttons-on-volunteer-page.spec.ts (11.4)
│   │   ├──click-all-buttons-on-donate-page.spec.ts (11.5)
│   │   ├──click-all-buttons-on-tos-page.spec.ts (11.6)
│   │   ├──click-all-buttons-on-privacy-policy-page.spec.ts (11.7)
│   │   ├──click-all-buttons-on-partnership-page.spec.ts (11.8)
│   │   ├──click-all-links-on-get-started-page.spec.ts (11.9)
│   │   ├──click-all-buttons-on-donation-thanks-page.spec.ts (11.10)
│   │   ├──click-all-buttons-on-about-page.spec.ts (11.11)
│   │   ├──click-all-buttons-on-about-foundation-page.spec.ts (11.12)
│   │   ├──click-all-links-on-about-oppia-footer.spec.ts (11.13)
│   │   ├──subscribe-newsletter.spec.ts (11.14)
│   │   ├──click-all-buttons-on-navbar.spec.ts (11.15)
│   │   ├──view-and-search-for-blog-posts.spec.ts (11.16)
│   │   ├──select-and-play-topic-from-math-classroom.spec.ts (12.1)
│   │   ├──browse-and-search-for-lessons-in-community-library.spec.ts (12.2)
│   │   ├──search-for-and-start-playing-a-collection.spec.ts (12.3)
│   │   ├──play-through-lesson-while-getting-feedback-and-hints.spec.ts (12.4)
│   │   ├──share-and-give-feedback-for-exploration-but-not-report-and-rate-it.spec.ts (12.5)
│   │   ├──choose-what-to-do-from-the-last-card-of-an-exploration.spec.ts (12.6)
│   │   ├──track-and-resume-exploration-progress-via-url.spec.ts (12.7)
│   │   ├──sign-in-and-save-exploration-progress.spec.ts (12.8)
│   │   ├──play-lesson-in-different-languages-and-listen-to-voiceovers.spec.ts (12.9)
│   │   ├──change-site-language-and-engage-with-original-exploration.spec.ts (12.10)
│   │   ├──use-keyboard-shortcuts-to-navigate-and-shift-focus.spec.ts (12.11)
│   │   ├──cannot-access-dashboards-which-require-login.spec.ts (12.12)
│   ├── logged-in-user
│   │   ├──create-and-delete-account.spec.ts (13.1)
│   │   ├──access-dashboards-and-other-pages-from-profile-menu.spec.ts (13.2)
│   │   ├──save-an-exploration-to-play-later.spec.ts (13.3)
│   │   ├──restart-or-continue-exploration-on-revisit.spec.ts (13.4)
│   │   ├──give-feedback-rate-and-report-an-exploration.spec.ts. (13.5)
│   │   ├──manage-goals-progress-and-lessons-from-learner-dashboard.spec.ts (13.6)
│   │   ├──subscribe-to-creator-and-view-all-explorations-by-that-creator.spec (13.7)
│   │   ├──edit-profile-preferences-and-export-their-account.spec.ts (13.8)
│   │   ├──set-language-to-rtl-and-navigate-through-site.spec.ts (13.9)
│   │   ├──edit-avatar-and-view-created-lessons-in-profile-page.spec.ts (13.10)
│   │   ├──use-keyboard-shortcuts-to-navigate-to-different-pages.spec.ts (13.11)
│   ├── translation-admin
│   │   ├── add-and-remove-translation-rights.spec.ts (9.1) (9.2)
│   ├── practice-question-admin
│   │   ├──add-and-remove-contribution-rights.spec.ts (9.3)
├── utilities
│   ├──common
```

```
│   │   ├── puppeteer-utils.ts
│   │   ├── show-message.ts
│   │   ├── test-constants.ts
│   ├── user
│   │   ├──blog-admin.ts
│   │   ├──blog-post-editor.ts
│   │   ├──curriculum-admin.ts
│   │   ├──topic-manager.ts
│   │   ├──mederator.ts
│   │   ├──site-admin.ts
│   │   ├──contributor-dashboard-admin.ts
│   │   ├──release-coordinator-admin.ts
│   │   ├──email-dashboard-admin.ts
│   │   ├──voiceover-admin.ts
│   │   ├──logged-in-user.ts
│   │   ├──logged-out-user.ts
│   │   ├──question-admin.ts
│   │   ├──super-admin.ts
│   │   ├──translation-admin.ts
├── data
│   └── blog-post-thumbnail.svg
```

Handling Console Errors and mobile support for Acceptance Tests :
While running acceptance tests, it is very much possible that some errors may appear in the console. These errors, while they might not cause a test to fail, could signal potential problems that need attention. They could indicate issues that, if left unresolved, might impact users. So, It's important to take care of console errors and log them properly while running the test so that developers can resolve them as well.

To capture these console errors, I'll utilize an event provided by Puppeteer that listens to the console message as the test runs and logs them. The event is page.on('console', msg => {...}).

However, it's worth noting that both the handling of console errors and support for mobile viewport have already been provided in a recent PR. This means that these features are already implemented and available for use in our tests.

Reviewing Existing WebdriverIO Tests :
After auditing the existing WebdriverIO tests, I have identified the Critical User Journeys (CUJs) that each test covers. This will help me understand the coverage of the existing tests and how they map to the user journeys. I've mapped the tests to the CUJs and documented the information in the table below. This table will serve as a valuable reference for future test development and will help me avoid duplication of tests. By "avoid duplication of tests", I mean that this mapping will allow me to see if a new test I'm considering writing would cover the same CUJ as an existing wdio test or a group of existing wdio tests. If it does, then I can decide which wdio test is to be removed once the new corresponding acceptance test is written, thus avoiding unnecessary duplication. This will save time and resources, and keep the test suite manageable and efficient. It will act as a guide, helping me understand which CUJs are covered by

which tests and will allow me identify any WebdriverIO tests that can be removed once a sufficient number of acceptance tests have been written.

No.	End to End (e2e) webdriverIO tests (suite name)	Critical User Journeys (CUJs) for which the acceptance tests are set to replace the corresponding end-to-end (e2e) tests.
1	Accessibility	Logged-out user(12.11), Logged-in user (13.11)
2	AdditionalPlayerFeature	Logged-out user(12.5, 12.6, 12.7)
3	adminPage	Site admin (4.1, 4.2, 4.3, 4.4)
4	Blog	Logged-out user – static pages (11.16)
5	checkpointFeatures	Logged-out user(12.5, 12.6, 2.7)
6	classroomPage	Logged-out user (12.1)
7	classroomPageFileUploadFeatures	Curriculum Admin (1.3)
8	collection	Logged-in user(12.3)
9	contibutorDashboardAdmin	Contributor dashboard admin(5.1, 5.2)
10	embedding	Logged-out learner – embedded lessons(14.1)
11	featureGating	Release Coordinator Admin (6.1, 6.2, 6.3, 6.4)
12	feedbackUpdates	Logged-in user(13.3, 13.5)
13	learner	Logged-out user(12.3, 12.5, 12.6, 12.7), Logged-in user (13.6)
14	learnerDashboard	Logged-in user(13.6)
15	Library	Logged-out user(12.2)
16	Navigation	Logged-out user – static pages (11.1 to 11.13)
17	PlayVoiceovers	Logged-out user(12.9)
18	preferences	Logged-in user(13.8, 13.9)
19	profileFeatures	Logged-out user(12.5, 12.6, 12.7), Logged-in user (13.10)
20	profileMenu	Logged-out user(12.12), Logged-in user(13.2)
21	publication	Logged-in user(13.3, 13.5, 13.6)

22	skillEditor	Topic Manager(2.8, 2.12)
23	subscriptions	Logged-in user (13.7)
24	topicAndSkillDashboard	Curriculum Admin (1.1), Topic manager(2.10, 2.11)
25	topicAndStoryEditor	Curriculum Admin (1.1), Topic Manager (2.2, 2.6, 2.9)
26	topicAndStoryViewer	Logged-out user (12.1, 12.5, 12.6, 12.7)
27	topicAndStoryEditorFileUploadFeatures	Topic Manager (2.2, 2.7)
28	Wipeout	Logged-in user(13.1)
29	Users	Logged-out user(12.10), Logged-in user (13.8, 13.9, 13.1)

The table above covers all the user journeys that need to be written before the corresponding WebdriverIO e2e-tests can be removed..

Improving Test Organization

Improving test organization is an important aspect of maintaining a robust and efficient testing suite. Here are some strategies that I'll be employing to enhance test organization:

Descriptive Naming: I'll use clear, descriptive names for tests and test suites to make it easier to understand what each test is supposed to verify. The name should clearly describe the functionality being tested or what is expected. I will also keep the names of the selector descriptive and crisp while making sure it remains consistent with the existing naming patterns.

Selection by Visible Text: In order to make our acceptance tests more reflective of the user's journey, I will select elements by the text that the user sees, wherever possible. This approach aligns with the user's perspective, as they interact with visible text and not with underlying selectors or IDs. This method ensures that our tests remain robust even if CSS or JavaScript changes, as they are not tied to specific functionalities or styles. It also enhances readability and maintainability, as the tests will mirror the user's actions more closely. I will create some custom functions to suit the needs.

Use of e2e- prepended selector only: In cases where where text selection is not feasible, I will use unique selectors as a fallback. I'll use e2e- prepended selectors for consistent and reliable element selection. These selectors offer a clear separation of concerns, ensuring that changes to CSS or JavaScript don't impact my tests, enhancing their robustness. They also improve readability and maintainability, making it evident that these selectors are meant for e2e tests, thus preventing inadvertent alterations by other developers. Unlike IDs, which are unique and often tied to specific functionalities, e2e- selectors are solely for testing, making them less brittle. This also means that I will need to assign e2e- prepended classes to elements wherever required.

Code Clarity over Comments: While comments can be useful in explaining things or crucial parts of the test, I see that the primary goal should be to make the tests themselves as clear and self-explanatory as possible. I will strive to write tests that are easily understandable on their own. However, in cases where

the code's purpose and implementation might not be immediately clear, I will comment to provide necessary context and explanation. This would allow others to understand, update or debug the test in the future effectively.

Consistent Structure: A consistent structure across tests can make them easier to read and understand. I'll make sure to follow the existing pattern of how tests are written and organized or I'll change the existing tests for the sake of consistency if I introduce some structural changes.
Directory Structure: We currently have a well-defined and easy-to-understand directory structure :

```
oppia/core/tests/
└── puppeteer-acceptance-tests
    ├── spec
    │   ├── blog-admin-tests (give user type name)
    │   │   ├── assign-role-to-users-and-change-tag-properties.spec.ts
    │   ├── blog-editor-tests
    │   │   ├── try-to-publish-a-duplicate-blog-post-and-get-blocked.spec.ts
    │   ├── logged-in-user-tests
    │   │   ├── click-all-buttons-in-about-foundation-page.spec.ts
    │   │   ├── click-all-buttons-in-about-page.spec.ts'
    │   │   ├── click-all-buttons-in-thanks-for-donating-page.spec.ts
    │   │   ├── click-all-buttons-on-navbar.spec.ts
    │   ├── practice-question-admin-tests
    │   │   ├── add-and-remove-contribution-rights.spec.ts
    │   ├── translation-admin-tests
    │   │   ├── add-translation-rights.spec.ts
    │   │   ├── remove-translation-rights.spec.ts
    ├── images
    │   └── blog-post-thumbnail.svg
    ├── puppeteer-testing-utilities
    │   ├── puppeteer-utils.ts
    │   ├── show-message-utils.ts
    │   ├── test-constants.ts
    │   └── user-factory.ts
    └── user-utilities
        └── blog-admin-utils.ts
        └── blog-post-editor-utils.ts
        └── logged-in-users-utils.ts
        └── question-admin-utils.ts
        └── super-admin-utils.ts
        └── translation-admin-utils.ts
```

However, as the project progresses and based on suggestions received from the mentor, I may modify this structure. In our testing strategy, currently, we organize the user journey tests or spec files based on the user role as demonstrated above.

Organizational Changes:

- Moving the existing puppeteer-testing-utilities under the utilities folder in a folder named common-utils. This is because these are after all utility files and are used across tests.
- Removing '-tests' from the folder in the spec directory and -utils form the files in the utilities directory. Given that all files under 'spec' are tests and all files under 'utilities' are indeed utilities, these suffixes might be redundant.

```
oppia/core/tests/
└── puppeteer-acceptance-tests
    ├── specs
    │   ├── blog-admin
    │   │   ├── assign-role-to-users-and-change-tag-properties.spec.ts
    │   ├── blog-editor
    │   │   ├── try-to-publish-a-duplicate-blog-post-and-get-blocked.spec.ts
    │   ├── logged-out-user
    │   │   ├── click-all-buttons-in-about-foundation-page.spec.ts
    │   │   ├── click-all-buttons-in-about-page.spec.ts
    │   │   ├── click-all-buttons-in-thanks-for-donating-page.spec.ts
    │   │   ├── click-all-buttons-on-navbar.spec.ts
    │   ├── practice-question-admin
    │   │   ├── add-and-remove-contribution-rights.spec.ts
    │   ├── translation-admin
    │   │   ├── add-and-remove-translation-rights.spec.ts
    ├── utilities
    │   ├──common
    │   │   ├── puppeteer-utils.ts
    │   │   ├── show-message.ts
    │   │   ├── test-constants.ts
    │   ├── user
    │   │   ├── blog-admin.ts
    │   │   ├── blog-post-editor.ts
    │   │   ├── logged-in-user.ts
    │   │   ├── logged-out-user.ts
    │   │   ├── question-admin.ts
    │   │   ├── super-admin.ts
    │   │   ├── translation-admin.ts
    ├── data
    │   └── blog-post-thumbnail.svg
```

Introducing Additional Functions:

I plan to implement a series of additional functions to write acceptance tests more effectively. These functions will be added to the **puppeteer.util.ts** file and may include:

- **clickAndWaitForNavigation(selector)**: This function will simulate a click event and wait for the page to navigate to the new URL. This would be useful for testing navigation and ensuring that the page is fully loaded.
- **getElementText(selector)**: This function will retrieve the text content of a specified element.
- **clickChildElement(parentText, childText)**: This function is to click on an element that contains the specific text especially when multiple elements have the same text. It takes two parameters: parentText and childText. The function first locates an element that contains the parentText. Once the parent element is identified, the function then searches within this parent element for a child element that contains the childText. Upon finding the child element, the function simulates a click event on this child element.

As the project progresses, I anticipate that additional requirements may emerge. In response to these evolving needs, I will continue to expand our suite of custom functions.

Fixing the e2e flakes or minor UI fixes needs as project proceeds:

Fixing e2e flakes: sometimes tests exhibit both a passing and a failing result with the same code. They can be quite challenging as they do indicate problems in the code, the testing environment, or maybe the test itself. When I encounter e2e flakes, my first step would be to report an issue and label it to the dev workflow team. In solving this I'll try to reproduce the flaky behavior locally and dig into the test logs and error messages. Once I've identified the cause, I'll implement a fix. This could involve modifying the test or, if the issue lies in the code being tested, updating that code. This ensures our tests remain reliable and accurately reflect the code's functionality.

Addressing minor UI fixes: Over time, there may be some minor UI issues that I may come across while fixing flakes or writing acceptance tests. These could be as simple as alignment issues, color inconsistencies, or more complex like component behavior issues. When such issues arise, I'll first confirm the issue, report an issue on our repo, and triage it properly. I may start working on a fix if it causes blockage in my process of writing acceptance tests. Depending on the issue, a fix might involve tweaking CSS, adjusting HTML, or modifying TypeScript. I'll also ensure to test the fix across screen sizes to ensure the issue is truly resolved.

Handling Long-Running Jobs

When writing acceptance tests, in our case, the only aspect that requires handling long-running jobs are Beam jobs. However, we can simplify this by utilizing existing Beam jobs that complete relatively quickly. For instance, we have the FindMathExplorationsWithRulesJob, which takes approximately 2-3 seconds to complete. This will allow us to test the Beam jobs without incurring undue delays.

Impact on Other Oppia Teams

Implementing acceptance tests would impact the LACE, dev-workflow, QA, and acceptance tests teams in the following ways:

Better Collaboration: Acceptance tests can enhance collaboration, particularly between the dev-workflow and QA teams. For instance, when the LACE team is developing a new feature, they can use acceptance tests to demonstrate the feature's functionality. These tests provide a clear description of the expected behavior, which can give the QA team a better understanding of the feature's input-output relationships.

Easier Onboarding: For team members new to the project, such as new members of the dev-workflow, LACE or acceptance tests team, acceptance tests can serve as a valuable learning resource. These tests provide concrete examples of how the system works, helping new team members familiarize themselves with the project more quickly.

Quality Assurance: The QA team can use acceptance tests to ensure that the system meets its requirements and functions as expected. This can help identify bugs early in the development process, before they affect users.

Confidence in Changes: Dev-workflow and LACE team, can benefit from the confidence that acceptance tests provide when making changes to the system. By running these tests, teams can ensure that their changes haven't inadvertently caused any issues, which can encourage more proactive code refactoring and system improvements.

Key High-Level and Architectural Decisions

Decision 1: Utilizing existing Beam jobs to test CUJs that involve running Beam jobs, such as the one in the release-coordinator tab. I have considered the following alternatives:

I have considered the following alternatives:

1. Creating a separate tiny Beam job specifically for testing purposes.
2. Using the existing Beam jobs for testing.

Among these, I believe that using the existing beam jobs make a better approach, because:

- The existing Beam jobs are already optimized and tested, reducing the risk of introducing new bugs.
- It reduces the development effort as there's no need to create and maintain a new Beam job.
- It allows us to test the system under conditions that are closer to real-world usage.
- Some existing Beam jobs complete in just 2-3 seconds, making them ideal for testing scenarios without causing delays due to long-running jobs.

The above approaches are contrasted in detail in the following table:

Green: Positive consideration
Orange: Mixed consideration
Red: Negative consideration

Criteria	**Alternative 1: Tiny Beam Jobs**	**Alternative 2: Existing Beam jobs**
Test Execution Speed	Fast, as the job can be designed to run quickly. However, this speed is	Varies, depending on the specific Beam job used. Some existing Beam jobs complete in just 2-3 seconds, making

Criteria	Alternative 1: Tiny Beam Jobs	Alternative 2: Existing Beam jobs
	theoretical until the job is actually created and tested.	them ideal for testing scenarios without causing delays due to long-running jobs.
Development Effort	Higher, as a new Beam job needs to be created and maintained. This includes not only the initial development but also ongoing maintenance to ensure the job continues to function correctly as the system evolves.	Lower, as the existing Beam job is already developed and tested. This reduces the risk of introducing new bugs and eliminates the need for additional development and maintenance effort.
Impact on Datastore	None, as the job can be designed not to affect the datastore. This ensures that testing will not interfere with other operations.	Depends on the specific Beam job used. However, since we're using an existing job, we can be confident that the datastore will remain isolated as test are to run either on CI or local development server. Additionally, I'll be executing the FindMathExplorationsWithRulesJob, which only reads from the datastore.
Risk of Introducing New Bugs	Higher, as creating a new Beam job introduces new code into the system, which could potentially introduce new bugs.	Lower, as the existing Beam jobs are already optimized and tested, reducing the risk of introducing new bugs.
Real-world Testing Conditions	Lower, as the job is designed specifically for testing and may not accurately reflect real-world usage conditions.	Higher, as the existing Beam jobs are used in real-world scenarios. This can provide more accurate and meaningful test results.

Risks and mitigations

Potential Risk	Mitigation
The existing job might take a long time to run, making the test slower and more difficult to manage..	We can select from existing jobs that can complete quickly. For instance, jobs like FindMathExplorationsWithRulesJob and GetCollectionOwnersEmailsJob typically finish in about 2 seconds.

Documentation changes

I'll list all the Critical User Journeys (CUJs) under each type of user and indicate which ones are covered by the acceptance tests. This will serve as a quick reference for anyone who wants to understand what aspects of the application are covered by the tests. I will do this in this spreadsheet and moreover in the release testing doc that consists of all the CUJs, as this is something the QA would be using.
Besides, in order to prevent release testers from repeatedly scanning the top-level directory, I will maintain and update the release testing document that encompasses all the Critical User Journeys (CUJs) to be tested. This will be done as new acceptance tests continue to come in throughout the coding period.

As the application evolves and new tests are added, I'll update the document and this wiki page to ensure it stays current. This way, it will continue to serve as a valuable resource for the teams and the developers onboarding.

Metrics Plan
Not required.

E2e testing plan

Not required

Implementation Plan

As I progress through the Acceptance Test project during GSoC, I anticipate that the time required for reviews will decrease. This expectation is based on the understanding that I will be continuously learning and improving throughout the project. With each task or PR merged, I will gain a deeper understanding of the codebase, the project's standards, and the expectations of my mentors. This will enable me to write higher quality code that aligns more closely with the project's requirements and conventions as the project continues.
Consequently, the need for extensive revisions should lessen over time, leading to more efficient and faster review cycles especially in the second milestone. This continuous learning and improvement is a key aspect of my approach to the milestones.

In the process of streamlining our development and review process, I have grouped related PRs together based on the functionalities they implement. This approach will help in maintaining a logical coherence in the codebase and also will facilitate a more focused and efficient review process. Importantly, these grouped PRs often share the same utility files or infrastructure. This shared infrastructure can include common setup code, shared constants, helper functions, or common testing utilities. By leveraging these shared resources, I can ensure a more consistent codebase, reduce the potential for errors, and enhance the maintainability of our project.

Community Bonding Period tasks.

1. **[Christie]**: Migrate all the acceptance tests to Jest. [Deadline: May 24th, for opening the PR]
2. **[Akhilesh]**: Make the organizational changes in the acceptance test directory and add additional functions. [Deadline: May 27th]

Milestone 1

Key objective for this milestone: The primary goal of this milestone is to develop acceptance tests that cover all Critical User Journeys(CUJs) experienced by admins. These tests would be run on both desktop and mobile viewports, ensuring that utility functions remain reusable in both cases.
Additionally, the process will involve addressing any minor UI/UX issues that are identified within these user journeys during the creation of the acceptance tests, and keeping the documentation on the wiki, CUJs doc and CUJ spreadsheet updated.

I have enumerated all the Critical User Journeys (CUJs) above, each assigned a unique Serial Number (Sr. No.). In the table below, please reference the appropriate Sr. No. to identify the CUJs being covered.

🟧 Orange: suites in which only certain tests will be removed and not the complete suite for it includes tests that fall under the CUJs of the editor and contributor, which are not in the scope of the project.

Sr. No.	Description of PR / action	Target date for PR creation	PRs in review, based on the date a PR is created.	Target date for PR to be merged	Wdio tests that would be removed in the PR
1	**Curriculum admin (1.1, 1.2, 1.3)** 1.1 : create-publish-unpublish-and-delete-topic-and-skill.spec.ts 1.2 : add-skill-to-diagnostic-test-and-subtopic.spec.ts 1.3: create-edit-and-delete-classroom.spec.ts ~ Mohd Afzal Khan will be covering this journey.	29 May	1	5 Jun	classroomPageFileUploadFeatures
2	**Moderator (3.1, 3.2)** 3.1: view-recent-commits-and-feedback-messages.spec.ts 3.2: edit-featured-activities-list.ts	31 May	1, 2	5 Jun	

3	**Site Admin (4.1, 4.2)** 4.1 : edit-user-roles.spec.ts 4.2 : load-dummy-data-in-dev-mode.ts	2 Jun	1, 2, 3	7 Jun	
4	**Site Admin (4.3, 4,4)** 4.3 : edit-platform-parameters.spec.ts 4.4 : use-misc-tab-features.spec.ts	4 Jun	1, 2, 3, 4	9 Jun	adminPage
5	**Release coordinator page (6.1, 6.2)** 6.1 : run-a-beam-job-and-copy-the-output.spec.ts 6.2 : edit-feature-rollout-configuration.spec.ts	6 Jun	3, 4, 5	12 Jun	
6	**Release coordinator page (6.3, 6.4)** 6.3 : flush-and-get-profile-of-redis-cache.spec.ts 6.4 : update-promo-bar-message.spec.ts	8 Jun	4, 5, 6	14 Jun	featureGating`
7	**Topic Manager (2.1, 2.2, 2.3)** 2.1 : create-publish-unpublish-and-delete-subtopic-and-story.spec.ts 2.2 : browse-topics-on-topics-and-skills-dashboard.spec.ts 2.3: filter-sort-and-open-a-skill.spec.ts	12 Jun	6, 7	19 Jun	

8	**Topic Manager (2.4, 2.5)** 2.4 : edit-and-preview-a-topic.spec.ts 2.5 : edit-and-preview-a-subtopic.spec.ts	14 Jun	7, 8	20 Jun	
9	**Topic Manger (2.6, 2.7, 2.8)** 2.6: edit-and-publish-a-story-only-with-mobile-supported-interactions.spec.ts 2.7: edit-preview-and-save-a-chapter.spec.ts 2.8: create-and-delete-questions-in-topic-editor.spec.ts	17 Jun	7, 8, 9	23 Jun	topicAndStoryEditorFileUploadFeatures
10	**Topic Manager (2.9, 2.10)** 2.9: create-and-delete-questions-in-skill-editor.spec.ts 2.10: create-merge-assign-and-delete-skills.spec.ts	19 Jun	8, 9, 10	23 Jun	topicAndStoryEditor,
11	**Topic Manager (2.11, 2.12)** 2.11: edit-and-republish-a-skill.spec.ts 2.12: cannot-do-curriculum-admin-actions.spec.ts	21 Jun	9, 10, 11	26 Jun	skillEditor topicAndSkillDashboard
12	**User browsing static pages (logged-out user) (11.1, 11.2, 11.6, 11.4)** 11.1 : click-all-buttons-on-contact-us page.spec.ts 11.2: click-all-buttons-on-creator-guid	23 Jun	11, 12	28 Jun	

	elines page.spec.ts. ~ Anthony Gallop Covering in the PR, which is already approved. 11.6: click-all-links-on-tos-page.spec.ts 11.4: click-all-buttons -on-volunteer-page.spec.ts				
13	**User browsing static pages (logged-out user) (11.5, 11.3, 11.7, 11.8)** 11.5: click-all-buttons-on-donate-page .spec.ts 11.3: click-all-buttons-on-teach-page. spec.ts 11.7: click-all-buttons on-privacy policy-page.spec.ts ~ Anthony Gallop is covering it In the PR, which is already approved. 11.8: click-all-buttons-on-partnership-page.spec.ts	25 Jun	11, 12, 13	28 Jun	

Milestone 2

Key objective for this milestone: The primary goal of this milestone is to develop acceptance tests that cover all Critical User Journeys (CUJs) experienced by learners, whether they are logged in or out.These tests would be run on both desktop and mobile viewports, ensuring that utility functions remain reusable in both cases.
Additionally, the process will involve addressing any minor UI/UX issues that are identified within these user journeys during the creation of the acceptance tests, and keeping the documentation on the wiki, CUJs doc and CUJ spreadsheet updated. .

I have enumerated all the Critical User Journeys (CUJs) above, each assigned a unique Serial Number (Sr. No.). In the table below, please reference the appropriate Sr. No. to identify the CUJs being covered.

Orange: suites in which only certain tests will be removed and not the complete suite for it includes tests that fall under the CUJs of the editor and contributor, which are not in the scope of the project.

No.	Description of PR / action	Target date for PR creation	PR under review	Target date for PR to be merged	Wdio tests that would be removed in the PR
1	**User browsing static pages (logged-out user) (11.13, 11.14, 11.5)** 11.13: click-all-links-on-about-oppia-footer.spec 11.14: subscribe-newsletter.spec.ts 11.15: click-all-buttons-on-navbar.spec.ts 11.16: view-and-search-for-blog-posts.spec.ts	9 Jul	1	14 Jul	Navigation Blog
2	**Embedded lessons (logged-out learner) (14.1)** 14.1: play-through-embedded-lesson.spec.ts	11 Jul	1, 2	15 Jul	embedding
3	**Logged-out user (12.1, 12.2, 12.3)** 12.1: select-and-play-topic-from-math-classroom.spec.ts 12.2: browse-and-search-for-lessons-in-community-library.spec.ts 12.3: search-for-and-start-playing-a-collection.spec.ts	15 Jul	3	21 Jul	classroomPage Library
4	**Logged-out user (12.4, 12.5, 12.6)** 12.4: play-through-lesson-while-getting-feedback-and-hints.spec.ts	18 Jul	3, 4	24 Jul	topicAndStoryViewer AdditionalPlayerFeature

	12.5: share-and-give-feedback-for-exploration-but-not-report-and-rate-it.spec.ts 12.6: choose-what-to-do-from-the-last-card-of-an-exploration.spec.ts				checkpoint Features
5	**Logged-out user (12.7, 12.8, 12.9)** 12.7: track-and-resume-exploration-progress-via-url.spec.ts 12.8: sign-in-and-save-exploration-progress.spec.ts 12.9: play-lesson-in-different-languages-and-listen-to-voiceovers.spec.ts	21 Jul	4, 5	27 Jul	PlayVoiceovers
6	**Logged-out user (12.10, 12.11, 12.12)** 12.10: change-site-language-and-engage-with-original-exploration.spec.ts 12.11: use-keyboard-shortcuts-to-navigate-and-shift-focus.spec.ts 12.12: cannot-access-dashboards-which-require-login.spec.ts	23 Jul	4, 5, 6	28 Jul	
7	**Logged-in user (13.1, 13.2)** 13.1: create-and-delete-account.spec.ts 13.2: access-dashboards-and-other-pages-from-profile-menu.spec.ts	26 Jul	5, 6, 7	31 Jul	Wipeout profileMenu
8	**Logged-in user (13.3, 13.4)** 13.3: save-an-exploration-to-play-later.spec.ts	28 Jul	7, 8	2 Aug	collection

	13.4: restart-or-continue-exploration-on-revisit.spec.ts				
9	**Logged-in user (13.5, 13.6)** 13.5: give-feedback-rate-and-report-an-exploration.spec.ts. 13.6: manage-goals-progress-and-lessons-from-learner-dashboard.spec.ts	30 Jul	7, 8, 9	4 Aug	Publication feedbackUpdates Learner learnerDashboard
10	**Logged-in user (13.8, 13.9)** 13.8: edit-profile-preferences-and-export-their-account.spec.ts 13.9: set-language-to-rtl-and-navigate-through-site.spec.ts	1 Aug	8, 9, 10	6 Aug	Preferences Users
11	**Logged-in user (13.10, 13.11)** 13.10: edit-avatar-and-view-created-lessons-in-profile-page.spec.ts 13.11: use-keyboard-shortcuts-to-navigate-to-different-pages.spec.ts	3 Aug	9, 10, 11	9 Aug	profileFeatures Accessibility
12	**Contributor dashboard admin (5.1, 5.2)** 5.1: manage-translators-and-reviewers.spec.ts 5.2: manage-question-submitters-and-reviewers.spec.ts	5 Aug	10, 11, 12	10 Aug	contibutorDashboardAdmin
13	**Email dashboard (10.1)** 10.1: create-queries-for-sending-bulk-emails.spec.ts	7 Aug	11, 12, 13	12 Aug	

During M2, I will manage to get quicker PRs, as the experience gained from writing acceptance tests in M1 will expedite the process and also I will be all available.

What if I miss a PR ?

If I were to miss the deadline for any PR of the project mentioned in the milestone during GSoC, here's how I would handle it:

Communicate: The first thing I would do is communicate with my mentors and the project maintainers. I would explain the situation, why the deadline was missed, and provide an updated estimate for when I expect to complete the work. I would be genuine and transparent with the things I am facing and will endorse open communication.

Allocate Extra Time: Recognizing that some tasks may take longer than anticipated, I would allocate extra time in my schedule as needed to compensate for the missed deadlines. This could involve working additional hours or re-prioritizing tasks to ensure that the deadlines are met.

Prioritize and Plan: Based on the feedback from the mentors and the project maintainers, I would prioritize the remaining work. If the missed deadline has caused a backlog of tasks, I would create a plan to tackle them efficiently. However, because of the backlog, I won't compromise with the ongoing deadlines.

Post-GSoC Commitment: In the event that I still miss a PR deadline and fail to get it merged in the GSoC period, I am committed to completing the PR even after the conclusion of GSoC . I understand the importance of the work I am doing and I am dedicated to seeing it through to completion.

Future work :

Upon the project's completion, I plan to focus on any Critical User Journeys (CUJs) that may remain unaddressed. This will include the CUJs for all users and any newly introduced features.

In addition to this, I am planning to implement some features like test retries, automatic screenshots at failure, etc. These features will help in improving the robustness of our testing suite and provide more information when a test fails, making it easier to diagnose and fix issues.

Furthermore, particularly, I want to ensure that Oppia delivers consistent performance across all devices and environments. I am genuinely intrigued by Oppia's workflow and am eager to learn from and contribute to the platform as part of the Oppia community.

Thank you!

Appendix for useful links.

1) Oppia Web QA tests matrix
2) Oppia Release testing doc
3) Oppia acceptance test wiki
4) Oppia GSoC'24 wiki

5) Puppeteer official documentation