Google Summer of Code 2022

“Helping learners when they get stuck”

Manan Rathi

Section 1: About You

What project are you applying for?

Helping learners when they get stuck.

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

Education is a basic necessity for every human and its significance cannot be stressed enough. Unfortunately, not all have access to it. Even those who are enrolled in schools might not be getting the necessary, effective education. Enter Oppia, a community-driven organization that aims to provide **high-quality** education that reaches millions of students in a sustainable way. Within the brief time that I’ve been contributing to this organization’s purpose, I have understood that Oppia strives to make learning engaging, effective and easily accessible. A solid example of the same is this amazing project itself.

Getting stuck while solving problems is quite natural. All of us have faced this issue, sometime or the other. In fact, I believe that struggling with a question, and finding the correct answer without any help from tutors or peers, is a great way to boost

confidence. It also helps develop real problem-solving skills, which is essentially the actual, deeper purpose of education.

At the same time, when a learner gets stuck on a problem, he/she might often feel helpless and lose the motivation to try again. So, we must ensure that we give the learner just the right amount of assistance necessary to get “unstuck”. It is essential that the learner *learns* the metacognitive skills to deal with “stuck-ness”.

“Helping learners when they get stuck” aims to remove many roadblocks that a learner might face while trying to solve a question. This project will go a long way in making the learning experience at Oppia more encouraging, engaging and effective. I want to work on this project because I’m genuinely interested in finding ways in which we can best address the learner’s confusion. I would strive to ensure that neither does the learner feel stuck nor does she always need to look out for assistance (Learned helplessness).

I love the fact that my contributions to Oppia are actually helping make a difference out in the real world. I really want to work on this project because it will make the core learning experience at Oppia much more enjoyable and fruitful. Working with Oppia is a valuable experience for me, and I’ve come to learn a lot of new things in this brief period. Not just a new tech stack, but also collaboration, writing clean code, working as a team, and more. This journey from a contributor to a member at Oppia has been an amazing, enriching experience.

Prior experience:

For the past 5 months, I have been actively contributing to Oppia and working with AngularJS, Angular 2+, Jasmine, Karma and Python. As a member of the LaCE-quality and Automated QA team for some months now, I’ve had a great experience fixing frontend bugs, server issues, and writing frontend and backend tests. I have gained sufficient knowledge regarding the parts of the Oppia codebase that are relevant for this project. In this brief period, I have raised 20 PRs (19 of which are merged) and 2 issues.
(*Last Updated: 31 March 2022*).

Project Experience with Oppia:

Earlier, the learners would lose their progress if they quit the exploration midway. Also, they had no way of tracking their progress through the exploration. Clearly, this was

really frustrating for the learners. The aim of the User Checkpoints project was to add support for checkpointing and implement numerous other additions.

I have implemented the entire frontend of the logged-in experience in the **User Checkpoints** (TDD) project headed by Sean Lip. I'll also be implementing the backend for the logged-out experience. This project has considerably improved my understanding of the codebase dealing with both the creator-end and learner-end frontend, and their integration with the backend.

Links to 5 Merged PRs in Oppia:

1. **Fix #11560: Add "Lesson Info" Modal.**
2. **Minor UI enhancements in Goals Tab on Learner Dashboard.**
3. **Fix parts of #14219: Increase backend coverage of two files to 100%**
4. **Fix parts of #12912: Enables no-else-continue and no-else-raise pylint checks**
5. **Fix part of #14187: Add frontend tests for `normalize-whitespace-punctuation-and-case.pipe.ts`**

Project size

Medium (~175 hours)

Project timeframe

Extended Timeline of June 13 - October 15.

Contact info and timezone(s)

Ph. No: +91 888-136-4466
Email: rathimanan27@gmail.com
Timezone: Indian Standard Time (GMT +5:30)
GitHub Handle: Manan-Rathi
Preferred Mode of Communication: Email, Google Hangouts, Whatsapp, Discord

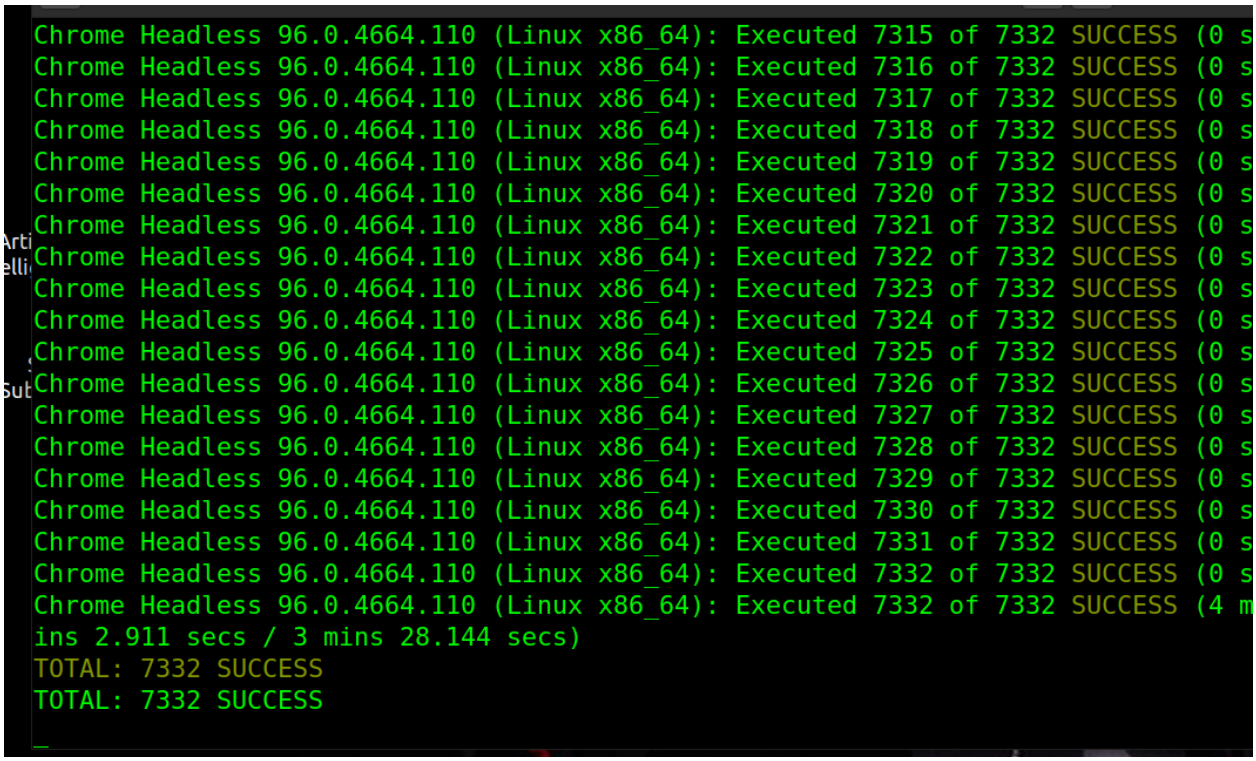
Time commitment

I'll be able to dedicate 20 hours/week for the project and thus, be able to complete it within the stipulated time period.

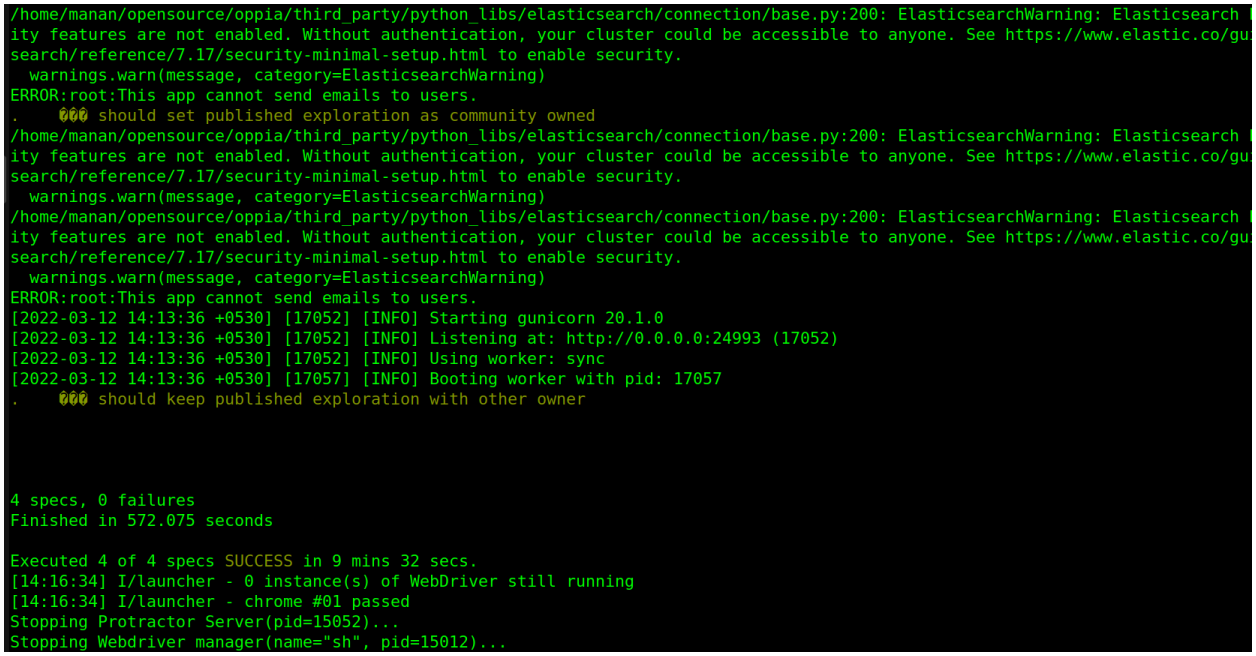
Essential Prerequisites

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

- I am able to run all the frontend tests at once on my machine.



- I am able to run one suite of e2e tests on my machine.



Other summer obligations

I'll only be applying for this project with Oppia. I'll be away for around 28 days in July-August. While I would still be working, my pace would be slower.

Communication channels

I am up to meet up with mentors on platforms such as:

- Discord
- Zoom
- MS Teams
- Google Meet

Alternatively, the mentors might suggest to me any other platform of their choice. I will try to notify the mentors about my progress on a daily basis (or as often as necessary) and will regularly be available on the platforms like Gmail, Hangouts, WhatsApp etc. I usually respond to Oppia-related notifications within 1-2 hours.

Section 2: Proposal Details

Problem Statement

Link to PRD (or N/A if there isn't one)	No Learners Get Stuck PRD [GSOC 2022]
Target Audience	Learners and Creators
Core User Need	As a learner, I need proactive and real-time help so that I can get unstuck even when external assistance might not be available. As a creator, I need a way to direct the learner to the linked concept card for refreshing their understanding, or provide them with adaptive feedback, or a different branch of states to offer more problem-focussed aid. The real-time assistance needs to be provided in a way that improves the learners' metacognitive skills on how to deal with "being stuck".
What goals do we want the solution to achieve?	• Allow creators to provide a destination state for the case where a learner is really stuck (which may or may not be a part of a short revision pathway). • Detect when a learner is stuck and provide appropriate real-time assistance based on redirection to concept cards, the alternative destination state, or proactive hinting. • Detect small misspellings and provide the learner with helpful feedback. • Prohibit lesson creators from sending the learner more than 2-3 cards back in the lesson.

Section 2.1: WHAT

Key User Stories and Tasks

#	Title	User Story Description (role, goal, motivation) *"As a ..., I need ..., so that*	Priority[1]	List of tasks needed to achieve the goal (this is the "User Journey")	Links to mocks / prototypes, and/or PRD sections that

[1] Use the **M**o**SC**ow system ("Must have", "Should have", "Could have"). You can read more here.

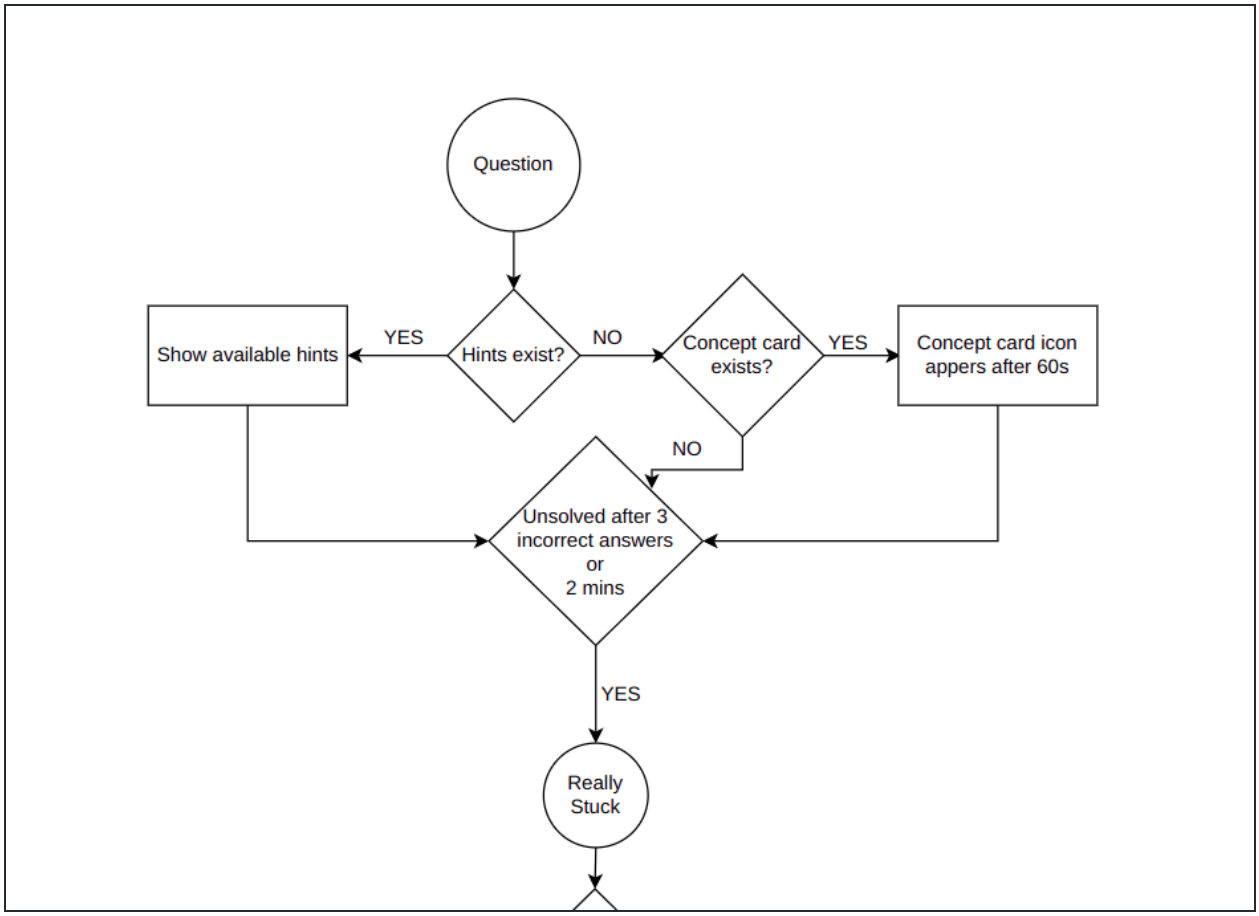
		….”			**spec out additional requirements.**
1	Detect when a learner is stuck and provide **appropriate real-time assistance.**	As a learner who is stuck, I need more meaningful assistance than hints or just asking me to retry.	Must have	The learner opens a question.	
				Learner is unable to solve the question within 60s. Gets first hint(if available) or a concept card(if available).	
				Learner goes through the first hint.	
				Learner does nothing for next 30s. Gets second hint.	
				Learner goes through second hint. Does nothing for 30s.	
				Learner clicks on the concept card icon and goes through the content. (If hints not available and concept card is linked)	
				Learner does nothing for 60s.	
				Learner sees an Oppia Response that prompts him to refresh his/her concept. Learner clicks on the continue button (if a dest_if_really_stuck state exists.)	
				Learner now sees another card. Learner revises the concepts.	
				Learner clicks on Continue.	
				Learner is directed back to the question.	
				Learner reattempts the question anew.	

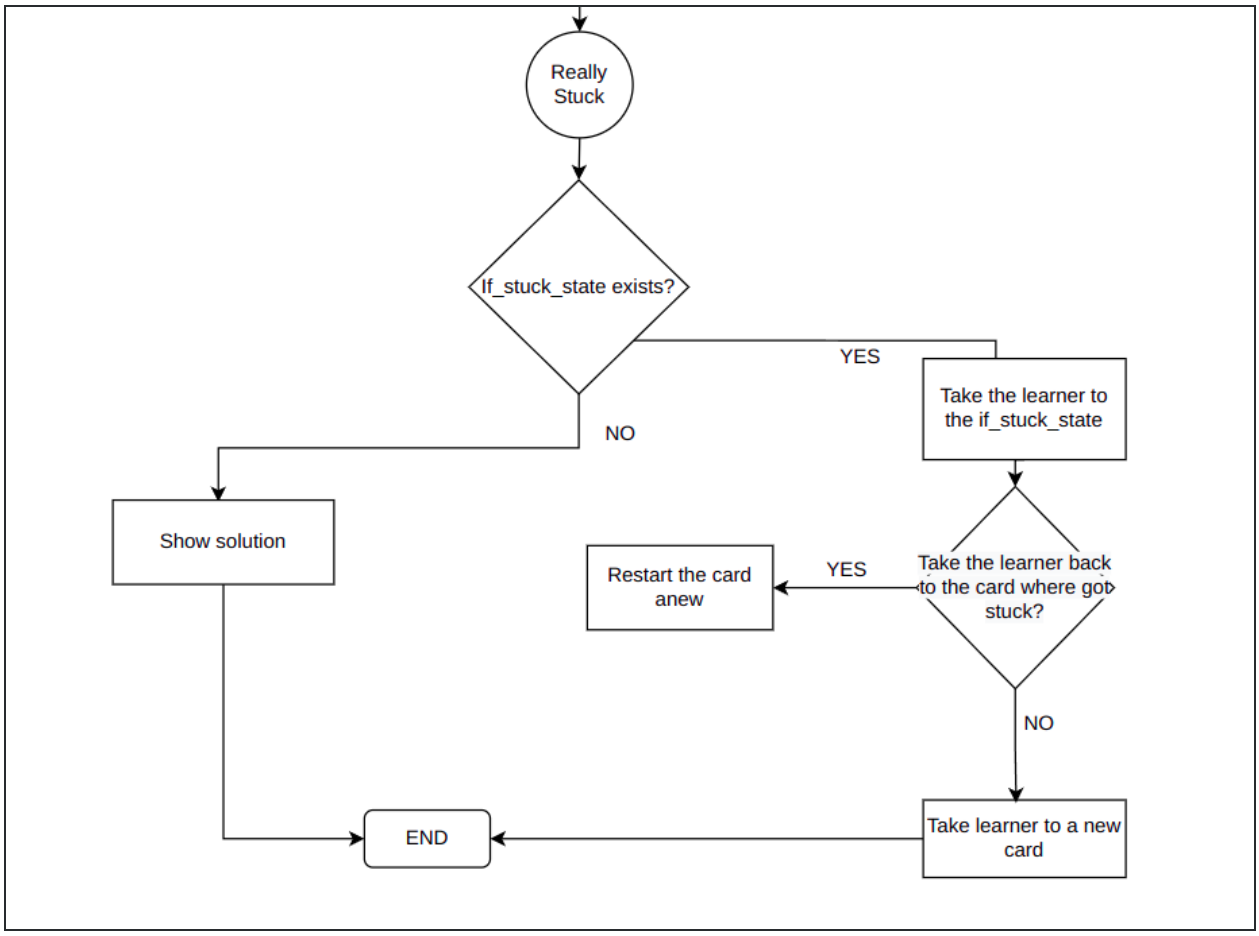
				Learner sees the solution button after making at least 3 incorrect submissions (after all available assistance is used).	
				The learner reads the solution, answers the question, and moves ahead.	
2	Allow creators to specify a destination state for the case when a learner is **really stuck.**	As a creator, I need a way to provide a destination state if a card needs a dest_if_really_stuck	Must have	Creator chooses an interaction.	Mock
				Creator should be able to fill in a destination state for the stuck learner. (May or may not be the part of a revision pathway)	
3	Prohibiting lesson creators from creating long “send-back paths”	As a learner, it is often discouraging to see loss of progress in a lesson. As a creator, **I need** to understand that I should not send learners too far back within the same lesson as a feedback mechanism since it can be really demotivating.	Must have	Creator sends the learner more than 2-3 cards back in the lesson. (In the exploration editor)	
				The creator tries to save the exploration.	
				A validation error is thrown, that has to be resolved before the exploration is saved.	
				The validation error prompts the creator to display the concept card or the new “if_stuck” state as an alternative to the long send-back path.	
4	Detect misspellings and provide helpful feedback. (For TextInput interactions)	As a learner, I am encouraged to retry a problem when I am prompted that my answer is close to the correct one, rather than just being told that my response is incorrect.	Must have	Learner types in an answer that has an edit-distance of 1-2 from the correct answer, and is not an exact match with the creator-defined wrong answer.	

				Learner is replied with something like "you're very close, you understand the concept, just check the spelling of the word XXX"	
				Also, for practice sessions and skill mastery, this response should be noted as a typo, and not as an error while recording stats in the backend.	
5	Ability to turn off the “catch misspellings” functionality on card-level	As a creator, I should be able to turn off the catch misspellings when some language-learning explorations may explicitly want to test/teach spelling.	Must have	A checkbox should be present to toggle the “catch misspellings” functionality.	

Final Proposed Implementation: (After incorporating suggestions from discussions & feedback)

Link to Doc showing the proposed learner flow: Proposed learner flow


Question
Hints exist?
YES
Show available hints
NO
Concept card exists?
YES
Concept card icon appers after 60s
NO
Unsolved after 3 incorrect answers or 2 mins
YES
Really Stuck



NOTE:

- Constants such as the number of incorrect responses and time limit that trigger the detection of stuck-ness will be easily parameterizable so that they can be easily tweaked if and when the need arises.

- This is merely the implementation idea that seemed most logical based on the feedback. **It is by no means rigid**. Obviously, this implementation is open to improvement if the need arises. For example, we can show the concept card before the hints if it turns out to be a more fruitful arrangement for the learners. At short notice, we could leverage a poll for the same.

- When a learner is stuck, it is extremely important that Oppia's response be carefully drafted. A few things that I would ensure while framing these responses are:

- Normalizing the idea of getting stuck – Use positive and encouraging language while helping the learner get unstuck.
- Ensure that they don't feel like being pushed around from one aid to another – We should let them know the **why** behind the specific aid.
- Ensure that we do not overdo the assistance – A little struggle is essential for cultivating metacognitive and problem-solving skills.

JUSTIFICATION:

- From the perspective of a "stuck" learner, it seems sensible to firstly receive **problem-focussed assistance** (**Hints**).

- If the hints are not available, we would direct the learner towards the linked concept card for refreshing his/her understanding. The **concept overview** offered by the card would help the learner if he/she had misunderstood or skipped some point in the related concepts.

- It doesn't really make sense to show both the hints and the concept card from the learner perspective because:
  - The material in hints and concept card is often similar.
  - Also, quite often the hints or responses direct the learner to the concept card themselves.

- If the learner is still stuck we can direct the learner to the "dest_if_really_stuck" state (which may or may not be the part of a short, dedicated revision pathway). This state could walk the learner through the concepts involved in the question.

- As the last resort, to make sure that there is an end to the learner's confusion, we would show the solution (correct answer along with suitable explanation) to the learner.

Technical Requirements

Additions/Changes to Web Server Endpoint Contracts

#	Endpoint URL	Request type (GET, POST, etc.)	New / Existing	Description of the request/response contract (and, if applicable, how it's different from the previous one)
1.	**explorehandler/init/**<exploration_id>	GET	Existing	This request is made by **ReadOnlyExplorationBackendApiService** to fetch exploration data from the backend using the exploration_id as the URL parameter. The response dict of this request will be modified to include two extra properties: dest_if_really_stuck (string) and catch_misspellings (boolean)
2.	**createhandler/init/**<exploration_id>	GET	Existing	This request is made by **EditableExplorationBackendApiService** to fetch the initial exploration data when the exploration editor page is opened using the exploration_id as the URL parameter. The response dict of this request will be modified to include two extra properties: dest_if_really_stuck (string) and catch_misspellings (boolean) which are to be initialized in the exploration-editor-page.

Calls to Web Server Endpoints

No new calls are needed to web server endpoints.

#	Endpoint URL	Request type (GET, POST, etc.)	Description of why the new call is needed, or why the changes to an existing call is needed
1.	**explorehandle**	GET	This call will fetch the exploration data on the learner's end. This

	r/init/<exploration_id>		data is needed to access the following: 1. dest_if_really_stuck (Exploration => State => Interaction => AnswerGroups => Outcome => dest_if_really_stuck) 2. catch_misspellings (Exploration => State => catch_misspellings) 3. linkedSkillId (Exploration => State => linkedSkillId) 4. Hints[ ] (Exploration => State => Interaction => hints)
2.	**createhandler /init/**<exploration_id>	GET	This call fetches the exploration object, so that the following new properties can be initialized: 1. dest_if_really_stuck (Exploration => State => Interaction => AnswerGroups => Outcome => dest_if_really_stuck) 2. catch_misspellings (Exploration => State => catch_misspellings)

UI Screens/Components

#	ID	Description of new UI component	I18N required?	Mock/spec links	A11y requirements
1.		A UI component to fill in destination state if stuck (if needed)	Yes	Mock	No
2.		A checkbox to turn on/off “catch misspellings” functionality in the exploration editor.	No	Mock	No

Data Handling and Privacy

No separate data needs to be collected.

Other Requirements

★ I18N for the new feedback in response to a stuck user (Notes on I18N)
★ All the new user-facing changes should be accessible and responsive.

Section 2.2: HOW

Existing Status Quo

Learners are currently provided assistance in the following ways:

- Hints (if they exist) are surfaced in the footer. The first one appears after 60s, and the subsequent ones after 30s.

- The lesson creator can direct the learner to a previous portion of the same lesson when they give incorrect answers.

- The lesson creator can help the learner brush his/her concepts by pointing them to a concept card for a “concept overview”.

There are some cons to the existing approach, which have been enumerated below:

- Even if the learner is really stuck, he/she is just prompted to try again, and not enough meaningful feedback is available apart from hints (if they exist).

- The offered feedback might often feel stale and robotic.

- Sending a stuck/confused learner quite back in the exploration can be really demotivating since they might feel discouraged by the loss in progress.

- Even if a learner merely misspells the answer (for **TextInput interactions**), he/she is just prompted to retry or potentially sent back to a previous part of the same lesson instead of being offered meaningful and encouraging feedback. So, a typo is just treated as a generic incorrect answer.

Solution Overview:

1. Allow creators to specify a destination state for the case when a learner is “really stuck”.

➔ How will the dest_if_really_stuck field be created and populated?

- The “dest_if_really_stuck” state is supposed to be a state that:

    1. Can point to the start of a separate pathway that walks the learner through the question.

    2. Or, can point to the start of a revision pathway that refreshes the learner’s understanding using a similar, simpler question or maybe offer him/her some hints.

- The field “dest_if_really_stuck” is added to the frontend and backend instances of the “Outcome” structure of “AnswerGroups”. For eg:

```
class Outcome {
 dest: string;
 dest_if_really_stuck: string | null;
 -------------Other fields---------------
}
```

- Since the “dest_if_really_stuck” is an optional field, it should take in **string or null**. So, the field remains “null” unless it’s populated from a UI component in the exploration editor page.

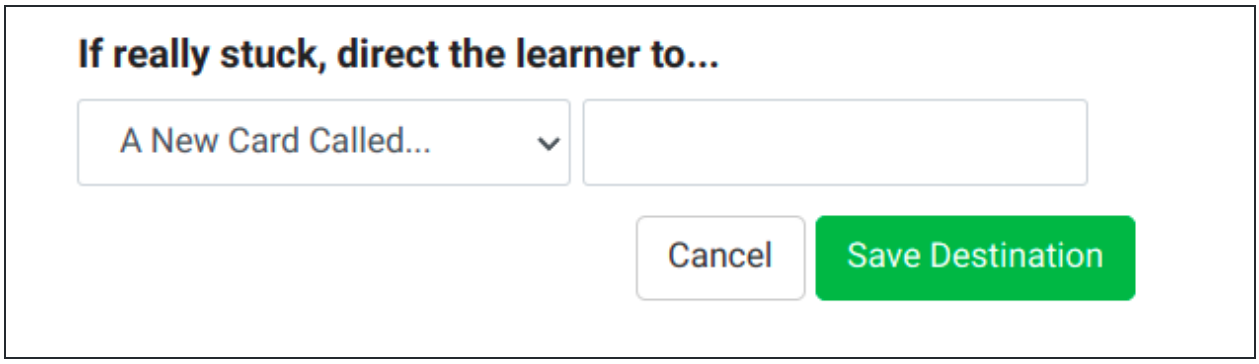
- For older explorations, the dest_if_really_stuck field will be set to null.

- We will create a UI component that comes up after the creator has chosen an interaction. This component would appear in two places since the answer group can be modified in: (Mocks for reference)
    - add-answer-group-modal

    - outcome-editor component

- Since there is a low probability of a learner getting stuck on questions that offer choices due to the limited number of options, the UI component is **not** active for such interactions **(Eg: Multiple choice interaction)**.

- This new component (`outcome-destination-if-stuck-editor`) would be placed **below** the existing outcome destination editor component. It will have a **form** to create a "dest_if_really_stuck" state, a "Save Destination" button, and a "Cancel" button:



- An info tooltip would be used to convey to the creators what the purpose of this field is. The message could be – "You can now specify a new card in which you can walk the learners through the concepts used in the question, if they get really stuck!"

- We could use `[(ngModel)]="outcome.dest_if_really_stuck"` to populate the "dest_if_really_stuck" field. Clicking on "Save Destination" would save this choice.

➔ How will a revision pathway be created?

- Once the creator creates a dest_if_really_stuck state, he/she can create a dedicated revision pathway by using the dest field of the State class, while providing hints and feedbacks along the way.

- At the end of the revision pathway, the creator can easily direct the learner back

to the source state by choosing the right dest.

- In my opinion, the number of cards should ideally be 2-3, since from the perspective of a really stuck learner, a long revision path doesn't really seem appealing.

➔ How will the dest_if_really_stuck state look in the exploration overview?

- In the exploration overview, the dest_if_really_stuck state will appear as a general node.

- We would use a **dashed arrow** to connect the state where the learner gets stuck and the "dest_if_really_stuck" state, as soon as the destination is saved (using the Save Destination button).

  We will use the `stroke-dasharray` attribute to specify the filled area and the unfilled area for creating dashed arrows.

2. Detect misspellings and provide helpful feedback (*For TextInput interactions*) & Ability to turn off the "catch misspellings" functionality on card-level.

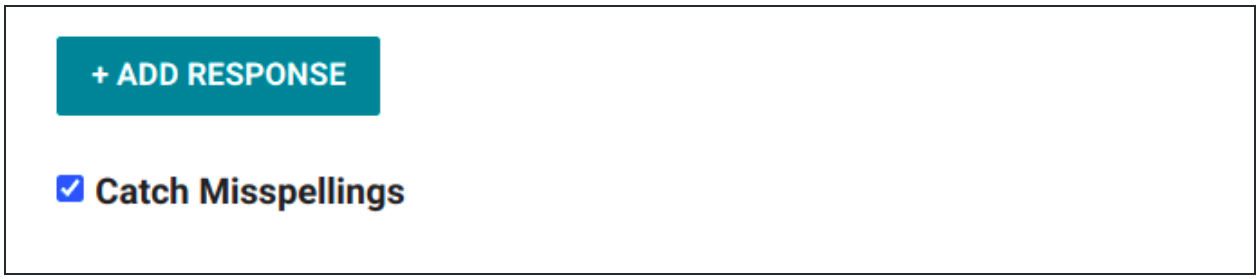
- Target: When the learner has an edit distance of 1-2 characters, he/she should be prompted to check the spelling of the word.

- Clearly, minimum_string_length_to_detect_misspellings could be 3 given that an edit-distance of 1-2 is permissible. This limit will be made easily parameterizable if the need arises to set a different minimum length. The constant for edit distance would be clearly mentioned in the feconf.py or constants.ts file so that it can be changed globally if needed.

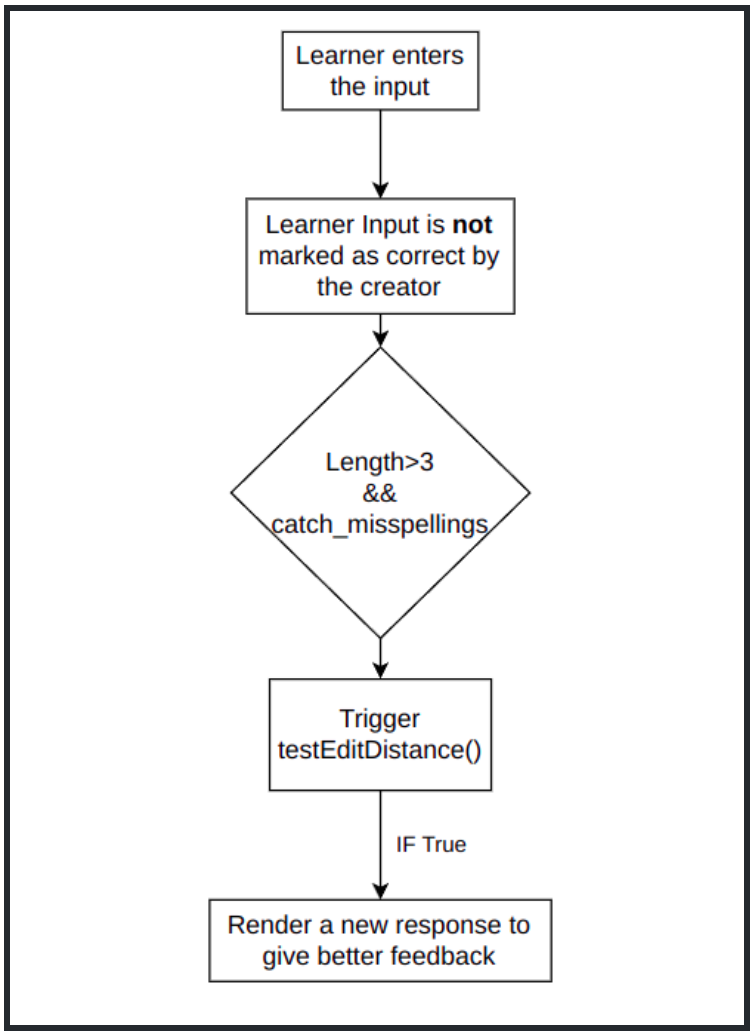
- For toggling the "catch misspellings" functionality on card-level:
  We will add a "catchMisspellings" field as a customization arg in the TextInput interaction.

The default value for new and older explorations would be **false** because even in TextInput interactions, the creator might not always need this functionality since some language-learning explorations may explicitly want to test/teach spelling.

This value could be toggled via a “checkbox” .It would look something like:



- We could check for misspellings in the following way:



- As soon as the learner submits the answer to a state, we check the **`answerIsCorrect`** field in the conversation-skin component. If the answer is incorrect, this means that this answer was not amongst the ones marked correct by the creator.

- We will further check if it matches with an input marked as **incorrect** by the creator. This can be done by iterating over the answer_groups and testing if the corresponding **outcome.labelledAsCorrect** is false.

- Now, if length of the answer > minimum_string_length_to_detect_misspellings (This is the minimum number of characters needed to catch misspellings i.e., 3) and if the catch misspellings field is true. The value of the latter can be easily found out from the associated state dict.

- Next, we will now check if the learner's answer was considered **incorrect** due to a spelling error.

- This can be done using a function `testEditDistance()`. This function will take in the learner's answer (TextInputAnswer) and inputs marked as correct by the lesson creator (TextInputRuleInputs).

A sample testEditDistance() function: Link to PseudoCode

```
testEditDistance(answer: TextInputAnswer, inputs: TextInputRuleInputs): boolean {
  const normalizedAnswer = this.nws.transform(answer).toLowerCase();
  const normalizedInput = inputs.x.normalizedStrSet.map(
    input => this.nws.transform(input).toLowerCase());

  const hasEditDistanceOneOrTwo = (
      inputString: string, matchString: string) => {
    if (inputString === matchString) {
      return true;
    }
    var editDistance = [];
    for (var i = 0; i <= inputString.length; i++) {
      editDistance.push([i]);
    }
    for (var j = 1; j <= matchString.length; j++) {
      editDistance[0].push(j);
    }
    for (var i = 1; i <= inputString.length; i++) {
      for (var j = 1; j <= matchString.length; j++) {
        if (inputString.charAt(i - 1) === matchString.charAt(j - 1)) {
          editDistance[i][j] = editDistance[i - 1][j - 1];
        } else {
          editDistance[i][j] = Math.min(
            editDistance[i - 1][j - 1], editDistance[i][j - 1],
            editDistance[i - 1][j]) + 1;
        }
      }
    }
    return (editDistance[inputString.length][matchString.length] === 1
      || editDistance[inputString.length][matchString.length] === 2);
  };

  return normalizedInput.some(
    input => hasEditDistanceOneOrTwo(input, normalizedAnswer));
}
```

- If we get an edit distance of 2 with any of the creator's inputs, we add a new response to render predefined feedback.

- It’s important that we don’t offer the same, mundane feedback every time the learner makes a typo. So, I propose creating an array of three I18N keys corresponding to three feedback statements (strings).

- A new function **getResponseToMisspellings()** could provide one of the three I18N keys randomly using the Math.random() function.

- These feedback responses would encourage the learner saying something like “You're headed in the right direction, but you need to recheck your spelling.”

3. Prohibiting lesson creators from creating long “send-back paths”.

➔ What will happen when a creator tries to send a learner back by 3 cards?

- After the creator has chosen a dest state in the outcome-destination-editor component, and he/she clicks on the “Save Destination” button, a BFS traversal over the states would be triggered.

- Let’s call this function: `bfsOverStates()`. This function will take in four arguments:

    1. `initStateName`: string *(The first state name of the exploration)*
    2. `states`: States *(All the states in the exploration)*
    3. `currentStateName`: string *(The state for which the destination is being set)*
    4. `destStateName`: string *(The destination state to which the creator wants to direct the learner)*

- After getting the destStateName and the currentStateName, the bfsOverStates() function will begin traversal from the destination state to the current state.

- While there exists a BFS traversal function **_computeBfsTraversalOfStates()** in the **compute-graph.service.ts** file, we cannot use it because we need to create a distance array to count the number of cards between dest state and current state.

➔ Why perform BFS over the states?

There are a few reasons to implement bfsOverStates() :

1. By performing BFS from the destState to the currentState, we ensure that a path exists from the dest state to the latter. This further implies that the dest state is indeed before the current state. (This is crucial since we **don't** have to restrict the creator from sending the learner **ahead** in the exploration by 2-3 cards)

2. Since a BFS traversal gives the shortest path between two nodes, we can calculate the minimum number of cards between the current state and destination state using a variable countOfCards (say).

- If the countOfCards exceeds 3, we'll raise a validation error in the exploration overview showing: "Such a long send-back path can be discouraging for the learner. Could you rather link a concept card or make a small revision pathway to help the learner get "unstuck" ?"

- This **critical** type warning must be resolved before the exploration is saved. We'll add the error message in the "app.constants.ts" file, and render it via the `exploration-warnings.service.ts` file.

4. Detect when a learner is stuck and provide appropriate, real-time assistance.

- Oppia already has an existing infrastructure to release hints that open up in a modal. The first hint is released after 60s, and the subsequent ones after 30s.

The proposed implementation for providing feedback integrates well with the existing infrastructure.

- We'll be using the existing modals for displaying the hints and the concept card.
- We could add a text just below the Hints section title for the creators to see when they try to add a new hint. The edge over adding the text below the Hints title would be that it wouldn't disappear when the creater starts typing.
- The creator would be advised to break the concepts involved in the question or maybe show how to solve a sub-problem in that question in a hint.
- The creator could be advised – "Provide a helpful hint to the learner. For example, you could show how to break down the question into smaller sub-problems, show how to solve a sub-problem, or relate this question to a previous one."

➔ How will we track the hints, incorrect responses and bind the responses with time?

- We fetch the number of hints by getting the length of the hints array. Also, we can keep a track of the number of hints consumed using the **numHintsConsumed** variable declared in **hints-and-solution-manager.service.ts**

- To get the number of incorrect answers submitted, we can simply use a variable **numberOfWrongSubmissions** in **conversation-skin.component.ts** that is set to 0 whenever a new card is initialized. Its value will be incremented whenever a wrong answer is submitted.

- Clearly, we need a way to release different aids at different time durations. For binding the numerous functions with time, we can use a **fixTimeout()** function. For example, this sample function replaces any queued timeouts, and sets the timeout for a new call.

```
fixTimeout(func: () => void, timeToWaitMsec: number): void {
  if (this.timeout) {
    clearTimeout(this.timeout);
  }
  this.timeout = setTimeout(func.bind(this), timeToWaitMsec);
}
```

➔ How will we nudge the learner towards the concept card?

- After the threshold for maximum incorrect answers or the maximum allotted time has been crossed, an icon appears in the footer. A tooltip appears to attract the learner's attention towards the card icon. This icon would stay even after the learner has viewed the concept card in case the learner wishes to view it again. It will disappear once the learner moves on to the next question.

- To check if the learner has already consumed the concept card, we can use a boolean **ConceptCardIsConsumed**. This boolean will be initialized with false. Once the **openConceptCard()** function is called, we'll set it to true. In this way, we will ensure that the concept card is provided only if it's **not** already consumed.

- The learner can open the concept card by clicking on the concept card icon in the footer, and can then dismiss the card using cancel, escape key press or a backdrop click after he/she has revised the concepts.

- We can get the linkedSkillId of a state from the "State" structure.

- Then, we can add a **(click)** attribute to the concept card icon that triggers a new **openConceptCard()** function inside **conversation-skin.component.ts**. Like:

```
openConceptCard(linkedSkillId: string): void {
  const modalRef = this.ngbModal.open(
    OppiaNoninteractiveSkillreviewConceptCardModalComponent,
    {backdrop: true}
  );
  modalRef.componentInstance.skillId = linkedSkillId;
  modalRef.result.then(() => {}, (res) => {
    const allowedDismissActions = (
      ['cancel', 'escape key press', 'backdrop click']);
    if (!allowedDismissActions.includes(res)) {
      throw new Error(res);
    }
  });
}
```

- It will be made mandatory for the creator to specify a solution for the state. In the end when all available assistance is exhausted, the solution is shown to the learner. This would be best since the learner would, in the end, get to know the right answer and the associated explanation. It will be shown to the learners **only if they have made atleast 3 incorrect submissions**.

  It can also be the case that the text fails in someway to convey the correct question. Having the solution option in this case could clarify the confusion and give the learners a way to move forward in the exploration.

- Regarding the other alternatives:

  - **Regarding allowing creators to pick and hope they do the right thing** – This would be the existing case. This could be a problem when the learner has exhausted all the assistance, and is still clueless about what's the correct answer. It is essential that, finally, the learner atleast gets the right answer with an explanation.

  - **Regarding allowing creators to pick and give them guidance on when and how to provide solutions** – In my opinion, this method would solve some of the issues like skipping providing solutions for easier questions and the test questions. However, the problem of when the learner gets really stuck and has exhausted all hints, could still persist in few cases.

  - **Regarding forcing the creator to classify the question into concept/test type** – Clearly, this method would solve the issue of keeping

solutions for concept ones and not the test questions but it would require a new field implementation. Also, this would increase the pressure on the creators to classify the question into one of concept/test. Alternatively,we could consider classifying explorations into concept-teaching and concept-testing ones instead.

- **Regarding keeping a separate exploration for the purpose of testing** – It would be a clean way to separate out the learning concepts part from the testing part into shorter, dedicated explorations. Since explorations are often long, this would prepone the satisfaction of learning a new concept for the learners.

  But clearly if the learners try to postpone the testing part, they'll definitely face issues in their concepts after a short while. We could perhaps provide a button that takes them to the testing exploration at the end of the concept exploration. The text associated with the button should persuade the learners that the testing part is short and important for enforcing their concepts.

- In the case that no if_really_stuck state exists and the learner has exhausted all the previous assistance, we would show the solution to the learner **if** the learner has made atleast 3 incorrect attempts (Implemented as a check so that the solution is not shown if they merely wait around for the hints).

- Also, if the if_really_stuck state exists and the learner is directed back to the state where he/she got stuck, we treat it as a fresh attempt and don't remember the number of hints the learner had used earlier. This is because we would need to remember what assistances the learner had used earlier on the state, which would stop making the experience memoryless.

Third-Party Libraries

No new third-party libraries are required.

"Service" Dependencies

These features add no new service dependencies.

Impact on Other Oppia Teams

The additional feedback being generated when a learner makes a typo will need to be translated. Other than that, there is no impact on any other Oppia teams.

Key High-Level and Architectural Decisions

Decision 1: Calculating the edit distance between the input and answer.

We have considered the following alternatives:

1. **Recursive solution:**
   The main idea is to process all the characters one by one starting from either the left or right sides of both strings. Let's consider a traversal from the right side. There are two possibilities for every pair of characters being traversed. Let's say m is the length of the first string and n is the length of the second string.

   - **If** the last characters of the two strings are the same, we count for the remaining string lengths. Thus, we recur for lengths m-1 and n-1.
   - **Else,** we consider the three mentioned operations on the last character of 'str1' and recursively compute the minimum cost of all three operations, and take the minimum value.
     ➔ Insert: Recur for m and n-1
     ➔ Remove: Recur for m-1 and n
     ➔ Replace: Recur for m-1 and n-1

2. **Dynamic Programming Approach;**

   In this approach, we still compute the minimum cost of the above three operations. Only this time, we avoid recomputing the same subproblems by constructing a temporary matrix that stores the results of the subproblems.

The above approaches are contrasted in detail in the following table. Clearly, 2 is the more efficient approach to calculating the edit distance.

	Alternative 1	Alternative 2
Time complexity	The time complexity of the 1st approach is exponential. We may end up doing $O(3^m)$ operations when no character of the two strings is common.	The time complexity of the 2nd approach is O(m x n).

Risks and mitigations

No potential risks are involved.

Implementation Approach

[Web only] Storage Model Layer Changes

None.

Domain Objects

- A new property is added to the “Outcome” structure:
    - dest_if_really_stuck: string

- A new property is added to the “State” structure:
    - catch_misspellings: boolean

User Flows (Controllers and Services)

User stories / tasks

- **Creator opens the exploration editor page for an exploration:**
    - ➢ A **GET** request is made to /createhandler/init/<exploration_id>
    - ➢ The response is an exploration object fetched from the backend.

➢ We use the fetched exploration data to get the following fields:
1. dest_if_really_stuck: string
2. catch_misspellings: boolean

➢ The fetched fields are initialized from the UI components.

- **Learner opens the exploration player page for an exploration:**

➢ A GET request is made to /explorehandler/init/<exploration_id>
➢ The response is the saved exploration data fetched from the backend.
➢ We use the fetched exploration data to make use of the following fields:
1. dest_if_really_stuck: string
2. catch_misspellings: boolean
3. linked_skill_id: string
4. hints: [ ]

[Web only] Web frontend changes

Subproject 1: Allow creators to specify a destination state for the case when a learner is “really stuck”

➢ Create a UI component for the creator to specify the dest_if_really_stuck state:

- A new **outcome-destination-if-stuck-editor.component.ts** file and **outcome-destination-if-stuck-editor.component.html** file is created inside **core/templates/components/state-directives/outcome-editor**.

- The new outcome-destination-if-stuck-editor component inherits the “outcome” object as an Input.

- The selector for the new component is placed inside the two template files from where the answer group can be modified –
  - outcome-editor.component.html
  - add-answer-group-modal.template.html.

- We will check that the interaction is not **MultipleChoice** type by using the getInteraction() function declared in **state-editor.service.ts**. This would decide whether the outcome-destination-if-stuck-editor component will be displayed.

- The output for the new component would be a new getChanges event emitter that is triggered via an updateChanges($event) function in the outcome-destination-if-stuck-editor component file.

- The purpose of outcome-destination-if-stuck-editor.html would be to:
  - Populate the dest_if_really_stuck state using the [(ngModel)] attribute. (Link to similar existing code).
  - Fire the updateChanges($event) function using the [(ngModelChange)] attribute.

➢ Represent the dest_if_really_stuck state in the exploration overview graph:

- In **state-graph-visualization.directive.ts**, we declare a new variable **augmentedStuckLinks** (say) that links states and dest_if_really_stuck states. It would store the source (current state) and the target (dest state).

- In **state-graph-visualization.directive.html**, we will iterate over the **augmentedStuckLinks** and use the `stroke-dasharray` attribute to create dashed arrow svgs. (Reference)

Subproject 2: Detect misspellings and provide helpful feedback (*For TextInput interactions*) & Ability to turn off the “catch misspellings” functionality on card-level.

➢ Create a checkbox to fill the catch_misspellings field:

- In **state-responses.component.html**, add a checkbox that initializes the value of the catch_misspellings field (A new field added to the State structure).

- Toggling the checkbox would call **onChangeCatchMisspellings()** in **state-responses.component.ts** using the **ng-change** directive.

- We could use a **new** data service – **state-catch-misspellings.service.ts** to store the displayed boolean value.

➢ Display a more meaningful response when the learner makes a typo:

- As soon as the learner submits an answer, submitAnswer() is triggered in the **conversation-skin.component.ts**. We use an already defined **answerIsCorrect** variable in the same file to judge if the learner's answer is correct.

- If the answer is incorrect but does not match with the creator-defined wrong inputs, we will call a new **testEditDistance()** function in **conversation-skin.component.ts** file.

- This function will take in the learner's answer (TextInputAnswer) and inputs predefined by the lesson creator (TextInputRuleInputs).

- **Pseudocode for testEditDistance():**

➔ Normalize learner's answer and creator specified correct inputs
➔ Iterate over creator specified correct inputs, pass answer and input to **hasEditDistanceOneOrTwo()**

- **Approach for hasEditDistanceOneOrTwo():**

➔ hasEditDIstanceOneOrTwo() returns the boolean true if the editDistance is 1 OR 2, and vice-versa.
➔ We will use this DP approach to create a 2D matrix editDistance[answer.length + 1][input.length + 1] (Reference)
➔ The value of editDistance[answer.length][input.length] gives the minimum edit distance.

- A meaningful response will be rendered using the addNewResponse() function declared in **player-transcript.service.ts**.

- To ensure that these responses don't feel robotic, one response would be chosen randomly at runtime from the following i18N keys:
  - "You're headed in the right direction, but you need to recheck your spelling."
  - "You're close to the right answer. Could you please correct your spelling?"
  - "Please recheck your spelling."

Subproject 3: Prohibiting lesson creators from creating long "send-back paths"

➢ Creator clicks on the "Save Destination" button after choosing a "dest":

- The button triggers a new function **bfsOverStates()** in **outcome-destination-editor.ts**. A variable countOfCards is declared to get the distance between the current state and the dest state.

- The function **sets the value of countOfCards** to the number of cards between the current state and the dest state (If a path exists).

- Parameters passed to **bfsOverStates()**:
  - initStateName – The name of the initial state.
  - states – The states' data present in the exploration structure.
  - currentStateName – The name of the **active state** in the exploration editor (The state for which the destination state is being set).
  - destStateName – The name of the state to which the creator wishes to direct the learner from the current state.

- We can get **initStateName** and **states** in the following way:

```
let initStateName = this.explorationInitStateNameService.savedMemento;
let states = this.explorationStatesService.getStates();
```

- **Pseudo algorithm of bfsOverStates():**
  - Get **stateGraph** using **_computeGraphData()** function declared in **compute-graph.service.ts**. The function _computeGraphData() takes in the **initStateName** and **states**.
  - Declare a new array **distance[ ]** that calculates the distance from the destState to all the other states. All elements in distance[ ] are initialized with 0.
  - Begin a generic BFS **from dest state**.
  - While iterating over all child nodes of the parent node: **distance[child] = distance[parent] + 1**
  - At the end of the traversal, set: **countOfCards = distance[currentStateName]**

★ Here, currentState is the name of the active state passed to the bfsOverStates function.

- We will declare a new boolean "**hasLongSendBackPath**" that will take **true** if the countOfCards > 3 and vice-versa.

- If **hasLongSendBackPath** is **true**, we push a **critical** type warning in the **_warningList** variable in the **exploration-warnings.service.ts** file with an appropriate message.

Subproject 4: Detect when a learner is stuck and provide appropriate, real-time assistance.

➢ Detect if the learner is stuck:

- The constants related to the threshold time limits and wrong answer limits would be declared in the **exploration-player-page.constants.ajs.ts** file. These constants can be:
  - MAXIMUM_INCORRECT_ANSWERS_BEFORE_CARD
  - MAXIMUM_INCORRECT_ANSWERS_AFTER_CARD
  - WAIT_BEFORE_CONCEPT_CARD
  - WAIT_AFTER_SECOND_HINT
  - WAIT_AFTER_CONCEPT_CARD

- We get the current state using the getState() function in the **exploration-engine-service**. From the fetched state, we can get the hints from the interaction. Then, depending on the number of hints, we trigger different sequences.

- We will use a new variable **numberOfWrongSubmissions** in the submitAnswer() function in the conversation-skin component. This variable will be incremented whenever the existing variable **answerIsCorrect** is false. This can be used to track the number of wrong responses submitted by the learner.

- We can bind the functions with time using the fixTimeout() function mentioned above.

➢ Make solutions mandatory for each state-card:
- Currently, the state-solution-editor modal appears when the learner has specified atleast one hint.
  - This can be changed by removing the condition `*ngIf="displayedHintsLength() > 0"`

- The solution would be made mandatory in two phases:
  - Phase 1 – (3 months): A info-warning would be displayed when the solution field would be empty.
  - Phase 2 – Add a *critical* type validation error would be thrown when the solution for the interactions has not been specified.

- The appearance of the solution icon has no major dependence on whether the hints have been consumed on the exploration-player side.

- The solution will only be shown to the learner 15 seconds after atleast 3 incorrect submissions were made and all the existing “helps” were consumed. A tooltip saying – “Stuck? You can take a look at the solution!” would appear at the same time.
    - This will be tracked using a variable totalWrongAnswersSubmitted that will be incremented whenever a wrong answer is submitted.

➢ Render the link for concept card as a response when the learner is stuck:

- We can get the **linkedSkillId** from the **linkedSkillId** field present in the state object.
- We can use the addNewResponse() function declared in **player-transcript.service.ts** to render a message nudging the learner towards the linked concept card.
- A new function **openConceptCard()** (Link to code) is declared in **conversation-skin.component.ts** for opening the concept card. This function takes in the linkedSkillId that is passed to **OppiaNoninteractiveSkillReviewConceptCardModalComponent** (The modal component that opens up to show the concept card).
- The concept card would be linked in the rendered message as an anchor tag that triggers **openConceptCard()** using the **(click)** attribute.

➢ Direct the learner to the dest_if_really_stuck state:

- Fetch the dest_if_really_stuck value (if it exists) from the outcome structure of the incorrect answer group.
- Check if all the other aids have been consumed using a boolean.
- Render a new response randomly from a list at runtime, using the addNewResponse() function declared in **player-transcript.service.ts**.

- The list of responses to render at runtime directing the learner towards the dest_if_really_stuck state –
  - “No, that’s not the correct answer. Let’s go through the concepts once again.”
  - Let’s take a quick revision of the concepts in this question.
  - You seem stuck. Let’s quickly go through the concepts once again!

- A continue button would appear.

- After the learner clicks on the continue button, direct him/her to the dest_if_really_stuck state.
  - Fetch the dest_if_really_stuck card using getStateCardFromName() declared in **exploration-engine-service.ts**
  - Use _addNewCard() declared in conversation-skin.component.ts and pass in the fetched card.
  - Once the conditions (All aids are exhausted & the learner hasn’t already visited the dest_if_really_stuck state) are met, show the newly added card.

Notes on I18N:

Currently Oppia carries out internationalization using distinct I18N keys for different strings. These keys are added to the assets/I18N/qqq.json file, and their translations are added to the respective language files. We carry out the translations dynamically using the translate pipe (in HTML) normally.

Which strings need I18N for this project?

- The 3 feedback responses offered to the learner when he/she makes a typo.
- The tooltip nudging the learner towards the concept card.
- The 3 feedback messages for directing the learner to the dest_if_really_stuck state-card.

How will the translations be carried out?

- I18N keys will be added for all of the above strings in the qqq.json file and the en.json file.

- In the template file, the translation will be done by adding the string “| translate” after the I18N key.
- For the translate service, after the service is injected into the component, the key needs to be passed into the *instant()* method of the service. This will return the translated string (if it exists).

Documentation changes

No documentation changes are necessary.

Testing Plan

E2e testing plan

We already have e2e test for :

- Testing the visibility of the hint icon and opening of the hints modal when the hint icon is clicked. (Link) (**Requires no tweaks**)

- Testing the visibility of the solution icon and opening of the solution modal when the solution icon is clicked. (Link)
    - **Requires tweak** – Adding in the condition that atleast three incorrect submissions are made

#	Test name	Initial setup step	Step	Expectation
1.	Creators can successfully specify a destination state for the case when a learner is **really stuck.**	Login and open the exploration editor page	The creator clicks on an answer group after choosing the interaction.	A new outcome-dest-if-stuck-editor UI component is shown.
			The creator fills in a dest_if_really_stuck state, and clicks on Save Destination.	A new node appears in the exploration overview.
2.	Learner can see the	Open exp player page	Learner is stuck for 60s and no hints exist but a linked	The concept card icon appears in the footer.

	concept card icon icon in the exploration player page when no hints are available.		concept card exists.	
			Learner clicks on the concept card button	The concept card modal opens up.

Cases to be tested:

- No hints or concept card or dest_if_really_stuck state – Just the solution appears.
- Only Hints exist – All the hints are shown, and the solution in the end.
- Only hints and a linked concept card exists – All the hints are shown, and the solution in the end.
- Only a linked concept card exists – Concept card is shown, and solution in the end.
- Only a dest_if_really_stuck state exists – The new state card is shown.
- Only hints and dest_if_really_stuck state exists – Hints are shown, and new state card is shown.
- Only concept card and dest_if_really_stuck state exists – Concept card and new state card are shown.
- All three exist – Hints and new state card are shown.

Karma tests:

All the changes to the component.ts and services.ts files will be accompanied by the associated tests in the respective .spec files. I will ensure 100% coverage of the changes done in the project.

Feature testing

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

Implementation Plan

Milestone Table – Milestone 1 (include both PRs and other actions that need to be taken prior to launch)

Milestone 1 – Allow creators to provide a destination state for the case where a learner is really stuck. Prohibit lesson creators from sending the learner more than 2-3 cards back in the lesson.

No.	Description of PR / action	Prereq PR numbers	Target date for PR creation	Target date for PR to be merged
1	Write state migration for dest_if_really_stuck field.	NA	1 July	9 July
2	• Create a UI component file and template file for populating the dest_if_really_stuck field. • Add an info tooltip for the dest_if_really_stuck field	1	10 July	17 July
3	• Make changes to state-graph-visualization.directive.ts and html file to create a new dashed-arrow link between dest_if_really_stuck state and the active state • Add frontend tests.	1,2	17 July	28 July
4	Create the bfsOverStates() function, and get the countOfCards between current and dest state.	NA	9 August	12 August
5	• Display a critical type warning with the appropriate message in the exploration-editor page • Add frontend tests.	4	16 August	20 August

–	Keeping one week free to address any arising project-related bugs	–	–	–

Milestone Table – Milestone 2 (include both PRs and other actions that need to be taken prior to launch)

Milestone 2 – Detect when a learner is stuck and provide appropriate real-time assistance based on (a) proactively showing concept cards or hints, (b) redirection to the alternative destination state, and (c) providing the solution as the last resort (the solution will be made mandatory for each state). Detect small misspellings and provide the learner with appropriate help.

No.	**Description of PR / action**	**Prereq PR numbers**	**Target date for PR creation**	**Target date for PR to be merged**
6	The catch misspellings checkbox appears on the editor page after state migration.	NA	26 August	31 August
7	On the learner side, the system is able to get the edit distance between the learner's answer and the correct answer.	6	29 August	5 Sept
8	Learner sees a meaningful message rendered when a typo is made	6,7	3 Sept	10 Sept
9	In the editor page: ● Solutions would be made mandatory, an info warning would appear when the creator doesn't specify the solution. ● Guidance to specify hints would appear as a text just below the hint section title.	NA	10 Sept	17 Sept

10	The learner will be shown the hints or concept card icon based on availability and time (with tooltips as needed). A tooltip will be added for the solution button.	9	25 Sept	4 October
11	Support to direct the learner to the if_stuck state is completed.	9,10,11	3 October	10 October
–	Remaining tests and bug fixes.	–	14 October	17 October

Future Work

1. Probably, the most critical aspect of detecting whether a learner is stuck and offering him/her real-time assistance is to get the timing right. After this feature is thoroughly tested, we could tweak the threshold constants to give the learners a more suitable amount of time and chances based on user studies.

2. To dis-incentivize the use of the solution option – Keep track of the cards where the learners used the solution option, and in the end **provide them with actionables on how to reinforce those specific concepts.**

3. **Consider a "report" button** – We could have a report button (linked to each state). This could help in reporting issues like "The question is unclear". And so, the question can be edited directly by the creator accordingly.

4. The tooltip associated with the concept card could be a new field that gives the learner an idea of what the concept card is actually about. So that, the learner is not disappointed when he/she clicks to find something that did not require revision in the first place.

5. We would add a critical type warning to cards (which would prevent the creaters from saving the exp if the solutions have not been provided), making solutions mandatory.

6. We could changes the icon for the solution button, since it's quite similar to the hint buttons.

Appendix: Summary of proposed solutions

Problem that the user faces	**Solution ideas**	**Ideal solution (and why)**	**What we'll actually implement (and why)**
Some questions don't have hints	1. Mandate hints for every question 2. Keep hints optional, but provide guidance to creators so they know how to write them.	Solution (2), i.e. provide guidance to the creator as follows: "Provide a helpful hint to the learner. For example, you could show how to break down the question into smaller sub-problems, show how to solve a sub-problem, or relate this question to a previous one." Solution 1 is less ideal because it would mean providing more assistance than is required (for questions that shouldn't need hints) and could lead to learners relying more heavily on hints for every question.	Same as the ideal solution (solution (2)).
Some questions don't have solutions	1. Mandate solutions for all questions. 2. Allow creators to pick and hope they do the right thing. 3. Allow creators to pick and give them guidance on when and how to provide solutions.	Solution 1 is ideal. Once the learner feels really stuck even after using all the available assistance, there should be an end to that stucked-ness. Also, in the case, when the learner fails to understand the question, there should be a way for him/her to move forward. Providing the solution (correct answer with the necessary explanation) is the way we can ensure both. We	Same as ideal solution (Solution 1) ● Make solutions mandatory for all cards. ● Show the solution only if the learner has made atleast 3 incorrect submissions

	4. Force creators to classify each question into one of two types (concept/test) and mandate solutions for concept questions. 5. Placing an upper bound on the number of times the solution can be viewed.	would also show the solution only when the learner has made atleast 3 incorrect submissions. Solution 2 is essentially the existing case. This would be a problem when the learner has exhausted the available assistance and is still clueless about the correct answer. Solution 3 would solve some of the issues like skipping providing solutions for easier questions and the test questions. However, the problem of when the learner gets really stuck and has exhausted all hints, could still persist in few cases. Solution 4 would allow the learners to keep solutions for the concept-teaching cards and maybe not the testing cards. But this would require an additional effort from the creators to classify each question. Solution 5 would clearly be a problem once the learner has already exhausted the number of turns to view the solution, and gets genuinely stuck on a question later.	and used all the available “helps”. ● Allow the creator to specify the solution without specifying the hints.
The learner fails to understand the question, or the question is unclear.	1. Showing the learner the solution (at the end as the last assistance since the	Solution 2 is ideal because this would allow the creator to directly review the question and make any changes if needed.	Currently, we would be implementing the Solution 1 since it would help the learners get unstuck from such questions.

	solutions were made mandatory for each question)as a way to get ahead in the exploration. 2. Add a “report question” button in the footer. So, that the learner can report the question, and the creator can directly review it.	Solution 1 would help the learner get past the question, and move forward (as proposed by making the solution mandatory for each question.)	Solution 2 is beyond the scope of the project.
The learner relies on the solution to get past the question.	1. Place a restriction that the learner can view the solution only if atleast 3 incorrect submissions have been made. 2. Keep track of the cards where the learners used the solution option, and in the end provide them with actionables on how to reinforce those specific concepts.	Solution 1 and Solution 2 together the ideal solution because it would ensure that the learner has made some attempts as well as best guide the learners on how to address the loopholes in their concepts.	Solution 1 because it would work as a temporary check in ensuring that the learner is shown the solution only if he/she has made some attempts to solve the question. Also, if several learners are using the solution button on a state, the creator could accordingly modify the text or the hints. Solution 2 is out of the scope of the project.