

# Building a lesson artist dashboard

Keshav Bathla

## Why are you interested in working with Oppia?

I found about Oppia in October 2018, when I was searching for organizations to start for Open source contributions, and from then on, have been a regular contributor to the Oppia platform. My initial reason to start contributing to Oppia was that I knew the languages used by the platform (python) well and so decided, this would be the best place to start with open source development. As I got involved in the project, I became more and more invested in the ideals of the Oppia Foundation, which is to provide a simple and easy to use learning platform in which anyone can share their knowledge about a subject to the world.

Among other things, one thing that I really liked about oppia is the explorations which is the main goal of oppia. I find the explorations very interactive in a way that they stimulate the natural thinking process which most schools lack today.

Also, the team at Oppia is very friendly. I have been warmly helped at each step of my contributions, whether be it with the code or the release testing.

## What interests you about this project? Why is it worth doing?

The platform of Oppia is undoubtedly a great platform for students to learn and for teachers (authors) to share their knowledge. I would love to improve anything that makes the work of a student or a teacher easier. The one thing that I love about this project is the image request. We all know as the creator of exploration can't be good in every domain like they can explain the topic well but when it's come to visualize a thing then some teachers may lack behind and we all know how important are the images for the exploration when it's come to visualize a thing. So to overcome this there will be an option which helps them to get images from all around the world without any restrictions. Implementing image request will play an important role because it will lead to increase collaboration in the exploration creation process.

## Prior experience (especially with regards to technical skills that are needed for the project)

I have worked on the following projects:

- Developed a website for gym freaks where they can read, create blogs and buy gym items.
- Text summarization. A tool that scrapes information from google based on your search keyword and return the summarized info which is relevant to the user.

- Debug and optimize a college technical festival website Vidyut.
- Other than this i have made some small projects like
  - Create a data entry and extraction tool for college technical groups in python gui library tkinter.
  - News at command line. A tool that gives news of your favourite channel at the terminal.
  - A chrome extension dictionary using tries and hash tables.
  - A tool that helps in playing a 3 patti game (not a game), while playing 3 patti players have to write and calculate info of each player but this tool helps in calculate of each player info and alert when user gets out.
- Except this while my major experience is in contributing to oppia.  
Member of following projects of oppia
  - Speed Improvements
  - Angular 2 Migration

Other than this, I have also participated in Delhi NCR hackathons like HACK CBS, DSC Hackathon, Hack With Tony, etc. and attend meetups like LinuxChix Meetup 2018.

I also do competitive coding, practise majorly on Hackerrank Platform and give some competitions on CodeChef, Codeforces, Hackerearth etc.

## Links to PRs to Oppia

- Allow editing a suggestion which is under review [#6368](#)
- Upgrade BeautifulSoup4 to its latest version [#6355](#)
- Refactor base.html to decrease the loading speed of pages by 2 seconds.
- Have some good discussion on some of the important issues like [#5714](#), [#6232](#), [#6158](#)
- Many more PRs for blocking bugs, issues, refactoring, etc. For a full list please click [Here](#)

## Reported issues

- Add progress bar at below the collection which were in progress in Learner Dashboard section [#6099](#)
- Audio bar not hiding for auto-generated audio [#6214](#).
- Views on exploration is not updating [#6583](#)
- Console error in audio translation mode [#6586](#)

- Checks marks are not aligned properly .

## Overview

Art and graphics form an integral part of most explorations on Oppia and are especially important for making lessons learner-friendly and communicating key concepts to learners, especially since good graphics can transcend language barriers. However, one major blocker when creating lessons is the lack of an easy way for designers/artists to add images to lessons. We would like to enable artists to contribute to lessons as easily as developers can contribute to GitHub repositories.

## Goal

We would like to implement a way to organize creation of graphics (using the existing suggestions framework) in a way that mimics the issue tracker on GitHub. More specifically, all open image requests should be surfaced to non-editors/artists of the exploration, who should have the option to suggest an image; once a suggested image is subsequently approved by the editors/artists, it can be added to the lesson.

## Steps involved while implementing this project.

### 1. Make images independent from state contents.

The first step is to make images independent from the state main content like audio, interactions are independent of state contents, like this, we should make images independent from state contents.

#### Why we are making images independent from state contents?

Currently, all our images info were inserted in HTML of main content with the tag `<oppia-non-interactive-image>`. We don't have any issue with this right now but we want in future that we can allow an artist to delete or insert new images in the exploration. And if we allow the artist to do actions on the image without this step then the image actions directly affect the exploration main content which can lead to some serious security issues.

**Note:** Before moving further one thing to say that in the current step there is the use of one thing called image-placeholder. What exactly image-placeholder is and how it will be inserted in exploration will be described in the next step so till now just assume it's like an image with some different configuration.

#### How should i implement this?

1. The first step is to make images independent from the state content like audio, interactions are independent of state content, like this, we should make images independent from state content.
2. In state Class, I should add one new field called "images\_assets".
3. Image\_assets object

```
"image_assets" = {
  "image_id_1" : {
    "src" : "<image_src>",
    "is_placeholder" : "<boolean_value>"
    "author_id" : "<author_who_updates_the_image_last_time>"
  }
}
```

#### Why we need image\_id ?

Now when we add image in the exploration the state content will be look like

```

<p> ..... </p>
<p> ..... </p>

<oppia-noninteractive-image></oppia-noninteractive-image>

<p> ..... </p>

<oppia-noninteractive-image></oppia-noninteractive-image>

<p> ..... </p>

```

There is no way to describe which image tag is for which image so to overcome this we add image id with the image tag and the content will be look like

```

<p> ..... </p>
<p> ..... </p>

<oppia-noninteractive-image id="<image_id>"></oppia-noninteractive-image>

<p> ..... </p>

<oppia-noninteractive-image id="<image_id>"></oppia-noninteractive-image>

<p> ..... </p>

```

Now we can determine for which image is the image tag.

### **Why we need is\_placeholder?**

To get to know which image object represent image or placeholder.

### **Why we need author\_id?**

There me sometime issues of licence on image. So author id is there to keep track which editor,creator or artist has updates images last time.

## **Process of adding image by creators or editors**

After clicking on image RTE component an object will be created

```

"image_id" = {
  "src" : "<image_src>",
  "is_placeholder" : "<boolean_value>",
  "author_id" : "<author_id>"
}

```

And a change object will be created

```

"change" : {
  "change_cmd" : "add_image",
  "state_name" : "<state_name>"
  "image_id" = {
    "src" : "<image_src>",
    "is_placeholder" : "<boolean_value>",
    "author_id" : "<author_id>"
  }
}

```

And appended in the change list.

And the function that applies this change command will append the image object in the image assets object.

### **What if the image is deleted?**

Whenever a change object for command add an image is created, it is not directly appended in the change list until and unless creator/editor has not saved the exploration. Change object will be appended in the temporary list.

So when creators/editor saves the exploration then each change object from the temporary list is taken out and check is the image id exist in new exploration or not. If not then the object will not be appended in change list and if exist then change object will be appended in the change list. And now the change list will be sent to the server for saving exploration.

As soon as the creators/editors clicks on image RTE component and upload images with details then the `<oppia-non-interactive-image id="image_id">` will be inserted in the state content and the image will be displayed to the creator.

### Summary

- An editor adds image and at the same time <oppia-non-interactive-image> tag will be appended in state HTML.
- After the above step, an image object and a change object will be created.
- After clicking on save draft button each change command will be applied.
- Now the exploration will get saved

So this tells that there is no change in inserting or displaying an image in client view. Client code will be changed for inserting images and placeholders in state content and for making images object.

## Coding implementation

### Backend

First step is to introduce one more field in state domain object of exploration and i.e ImageAssets.

- To do this create class ImageAssets( ) in file core/domain/state\_domain.py with having following functions
  - \_\_init\_\_( ) : For initializing an image assets object.
  - validate( ) : For validating the image assets object.
  - from\_dict( ) : Function for creating image assets object from image assets dictionary.
  - to\_dict( ) : Function for creating image assets dict from image assets object
- Along with this we have to edit class State for adding one more variable image assets. And also add one more function update\_image\_assets( ) for updating image assets.
- Edit some already created test for exploration because adding one more field leads to failing tests.
- Edit validation of state function.

### Files to be modified :

- core/domain/state\_domain.py
- core/domain/exp\_domain.py

### Test files

- core/domain/state\_domain\_test.py



- core/domain/exp\_domain\_test.py
- core/storage/exploration/gae\_models\_test.py

## Frontend

Now we have to populate each image tag with the image info like source, caption and description, so after taking each image id we have to find each image object and store the image info in the respective directive scope.

File to be changed :

- core/templates/exploration\_editor/ExplorationStatesService.js
- core/templates/exploration\_editor/ExplorationSaveService.js
- core/templates/exploration\_player/ExtractImageFileNamesFromStateService.js
- extensions/rich\_text\_components/image/directives/OppiaNoninteractivelma  
geDirective.js
- core/templates/dev/head/pages/exploration\_editor/ChangeListService.js (for adding a new type of change object in exploration change list)

Other thing to do in this step is the state migration.

State migration is divide into 2 steps

1. Migration job.
2. One-of job.

## Migration job

In this i do the state migration like

- Make the necessary changes to the NEW\_STATE\_TEMPLATE in the constants.js file to reflect the post-migration state dict structure.
- Increment the CURRENT\_STATES\_SCHEMA\_VERSION in the feconf.py file.
- Increment the CURRENT\_EXP\_SCHEMA\_VERSION in the exp\_domain.py file and similar changes in the question\_domain.py file.
- Start with writing  
\_convert\_states\_v(old\_state\_version)\_dict\_to\_v(old\_state\_version + 1)\_dict  
method in exp\_domain.py files under Exploration class and in  
question\_domain.py under Question class.
- Change the dict and yaml form of state in the following files wherever required:
  - core/controllers/editor\_test.py

- core/domain/exp\_domain\_test.py
- core/domain/exp\_services\_test.py
- core/domain/question\_jobs\_one\_off\_test.py
- core/domain/QuestionObjectFactorySpec.js
- core/domain/state\_domain\_test.py
- core/tests/test\_utils.py (Change the VERSION\_(Old\_version)\_STATE\_DICT to a new one)

### **One-of job**

Why I have to do this?

There is some editing of data models like while introducing one more field(i.e image\_assets) for the exploration. So adding one more field in data model leads to create null value for the newly created field in older explorations data and we also

So to overcome the above problem I will have to write one of the jobs.

How should i do this?

Adding one more class method in exp\_domain.py which accepts exploration data in dictionary format. This method makes image assets field to be an empty dict. Along with these i write functions that extract all images from the exploration and create an image assets object and then store it in image assets field.

## 2. Add image placeholder feature.

### What is image placeholder?

Image placeholder defines a space of an image in an exploration card that will be used by the creator to give a hint to a community about an image is needed here in the exploration.

### Why do we need this ?

- While creating exploration, creator thinks that there is a need for an image but currently he/she don't have it.
- So creators open an image request for the exploration. This exploration is visible in the contributions dashboard page.
- Now when the community wants to suggest an image they didn't get where is the need for an image in the exploration.
- So to overcome this creator adds image placeholder in exploration that defines space of an image in exploration and gives a hint to the community about an image is needed here

### Architecture of Image placeholder

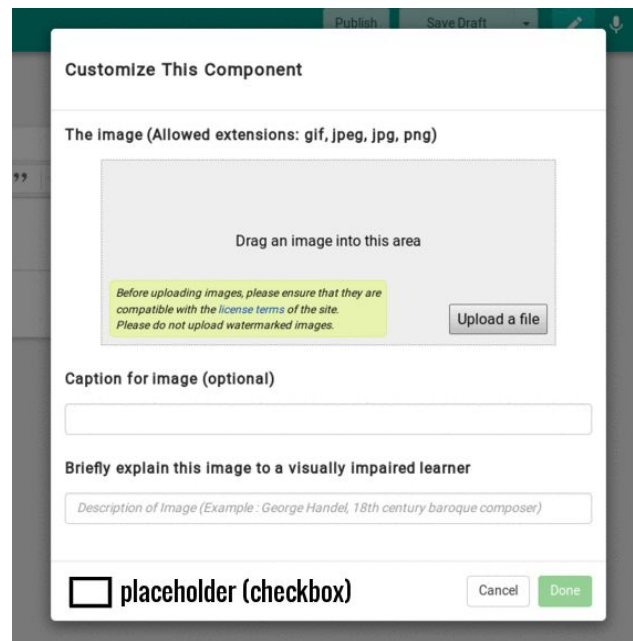
Image placeholder template should be look like this in exploration card



## How should be this implemented ?

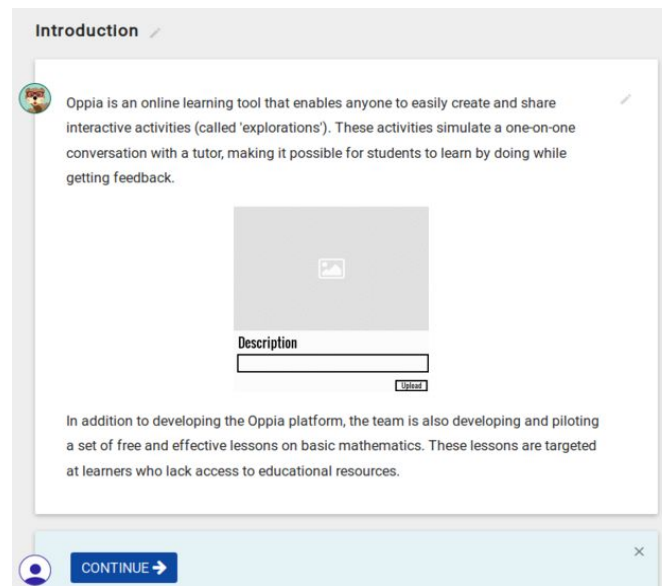
By modifying the existing image RTE component.

1. First creator clicks on image plugin button for inserting image or image-placeholder.
2. Then respective schema form will open having checkbox for placeholder on it.



The screenshot shows a 'Customize This Component' dialog box. At the top, there are tabs for 'Publish' and 'Save Draft'. The main section is titled 'The image (Allowed extensions: gif, jpeg, jpg, png)'. It contains a large gray area with the text 'Drag an image into this area'. Below this area is a yellow box with the text: 'Before uploading images, please ensure that they are compatible with the [license terms](#) of the site. Please do not upload watermarked images.' To the right of this box is an 'Upload a file' button. Below the image area is a section titled 'Caption for image (optional)' with a text input field. Underneath that is a section titled 'Briefly explain this image to a visually impaired learner' with a text input field containing the placeholder text 'Description of Image (Example : George Handel, 18th century baroque composer)'. At the bottom of the dialog, there is a checkbox labeled 'placeholder (checkbox)'. To the right of the checkbox are 'Cancel' and 'Done' buttons.

3. Then on checking placeholder option, and clicking on done button the exploration card will be look like this.



## Some technical details of change commands

Table describing which change commands created on adding image or placeholder

On adding	edit_state_content	add_image
placeholder	Yes	Yes
image	Yes	Yes

## Code of add\_image, delete image and update image command

```
def add_image(self, image_id, image_info):
    """Adds default image object in state.

    Args:
        image_id: str. ID of an image.
    """
    self.image_assets_mapping[image_id] = {}
    self.image_assets_mapping[image_id]['src'] = image_info['src']
    self.image_assets_mapping[image_id]['is_placeholder'] = image_info['is_placeholder']
    self.image_assets_mapping[image_id]['author_id'] = image_info['author_id']

def delete_image(self, image_id):
    """Deletes an image from the state.

    Args:
        image_id: str. ID of an image.
    """
    del self.image_assets_mapping[image_id]
```

These codes will be written in **core/controllers/state\_domain.py**

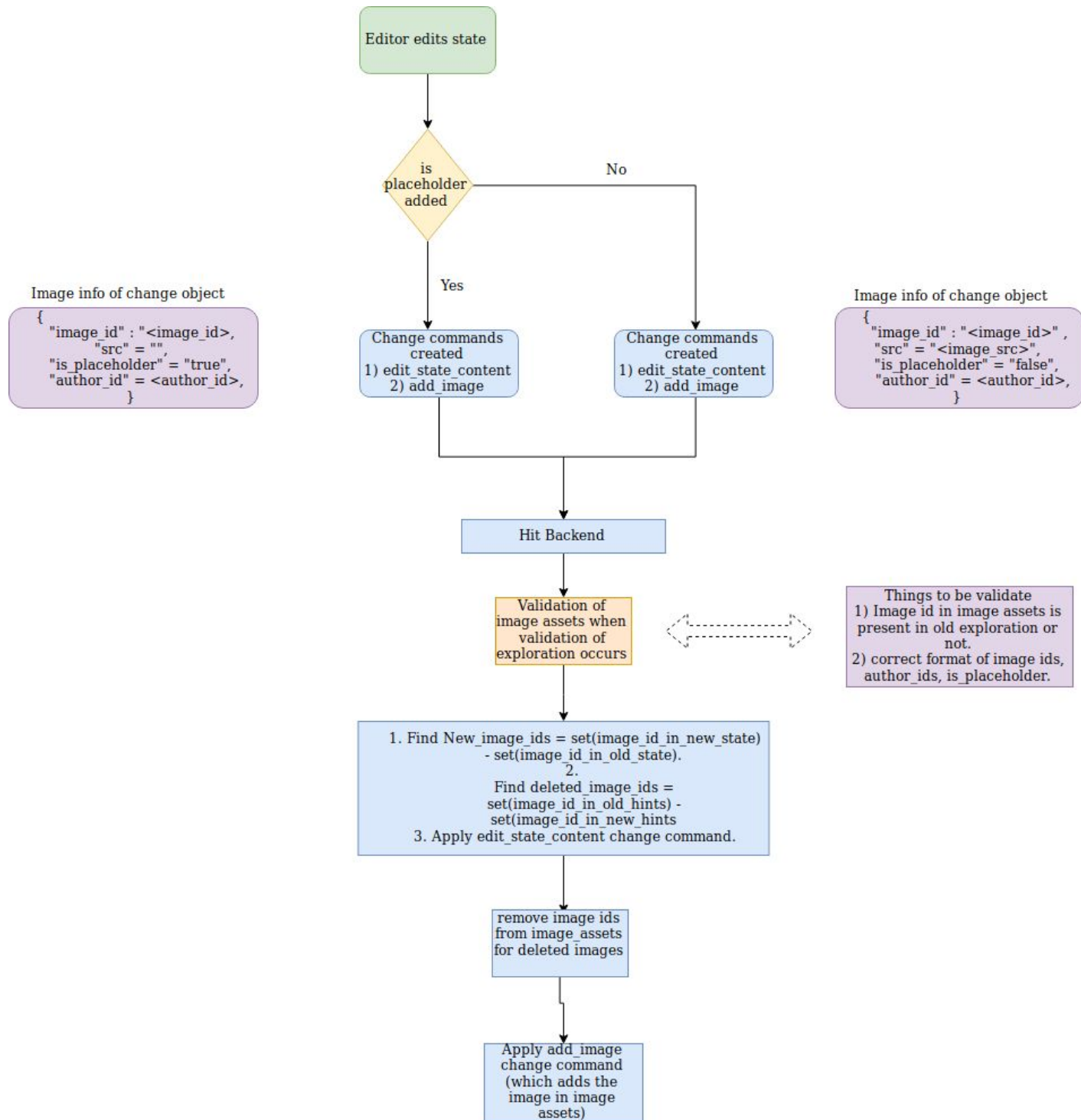
**Things to validate**

1. Image Id in image assets is present in old exploration or not.
2. Validate did all image\_ids in image\_assets old exploration are unique or not.
3. Check format of image\_id, author\_id.
4. Check is is\_placeholder boolean or not.

**Other things to validate**

1. Validate that <oppia-non-interactive-image> doesn't contains src, author\_id and is\_placeholder in state content html.
2. Validate that <oppia-non-interactive-image> should contain only contain image\_id, caption, description.
3. Validate that ( no. of new image ids == no. of change objects having change cmd 'add\_image')

## Flow of adding image placeholder and applying change commands



### Some questions and its answers

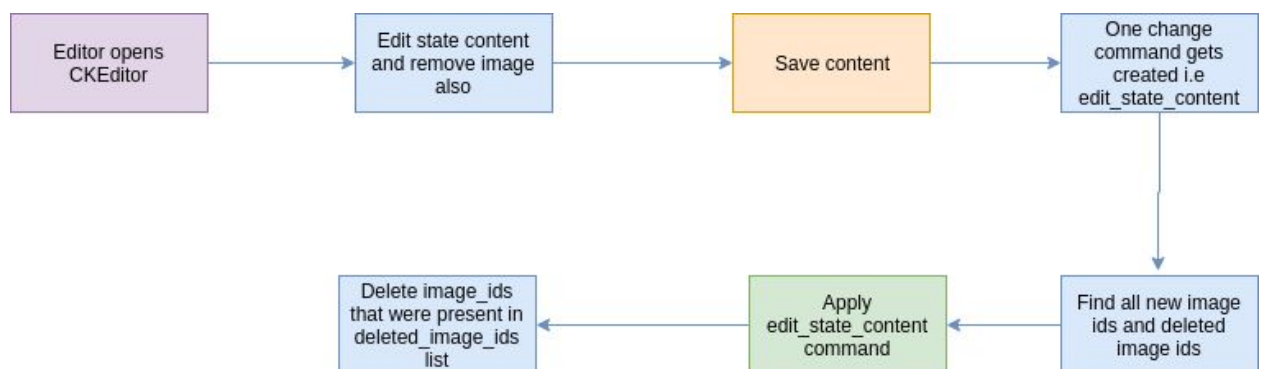
#### **How to find old and new image ids?**

Old and new image ids should be find with the help of regex and some string manipulation.

#### **How we are going to delete the image?**

For deleting an image, there is no such specific command. Like earlier, creators go to exploration editor opens ckeditor and removes image and saves the exploration. Now also, creators do the same process for removing/deleting the image. And when edit\_state\_content change command runs it gets all the deleted image\_ids and remove them from the image assets.

Flow diagram



#### **How will you make sure that the image\_id present in the change command is correct?**

Before adding image in image assets it should be checked, is the image\_id present in new\_image\_ids list or not. And if not error will be raised ("Image id does not exist")

#### **Can an exploration be published with placeholders?**

No, the exploration cannot be published with placeholders because it doesn't make any sense to publish explorations with having placeholders in them. If the exploration is published and let suppose learner visits the exploration so while playing the exploration he/she gets placeholders in the exploration which makes the learner confused.



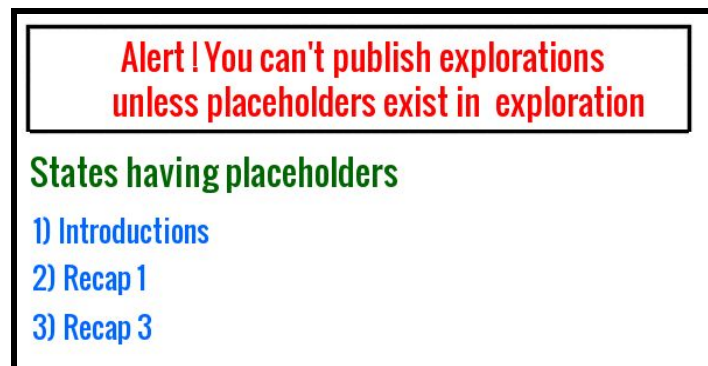
### So what if there were placeholders present in exploration and the creator try to publish an exploration?

When creators click on the publish button, then a function will run to check is placeholder present in the exploration.

- If no then the exploration gets published.
- If yes then a warning box will appear showing the exploration card names which were having the image placeholders in their content.

**Note :** The function will check is there any object in image assets where placeholder exists.

Warning box



(Image is not pixel perfect but it gives the brief idea about how the warning box will be there)

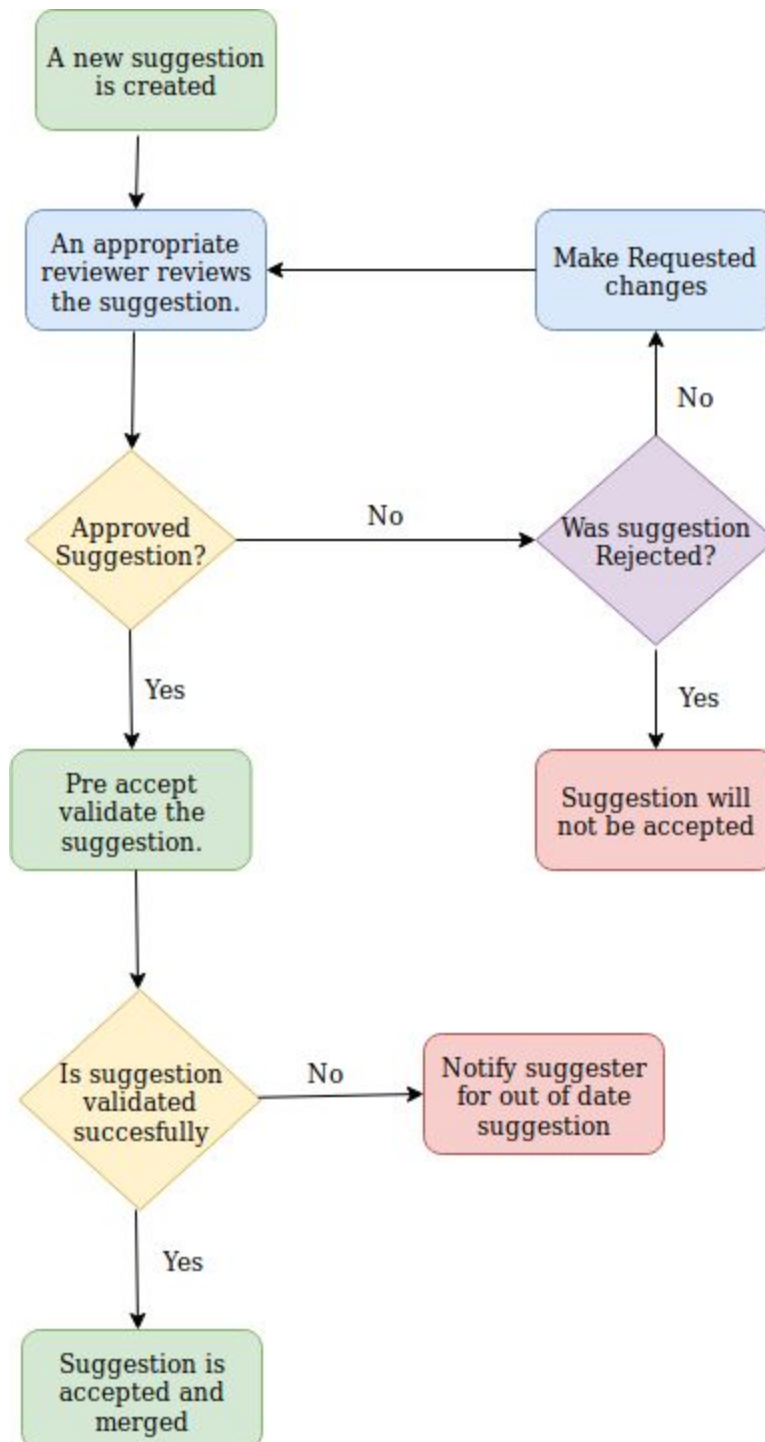
### Coding implementation

- Convert image RTE to plugin. This is to be done because we have to add dropdown menu and which can be implemented in plugin not in RTE component.
- Files to be modified:
  - extensions/rich\_text\_components/Image/directives/ImageDirective.js
  - extensions/rich\_text\_components/Image/protactor.js
  - extensions/rich\_text\_components/Imagedirectives/image\_directive.html

Till now creator is adding images and placeholders in exploration. For need images in place of placeholders, creators/editors will open image request for the need of an image. So community who wants to contribute will go to the community dashboard page for contributing image. There they will get the list of explorations which needs an image. And from there community suggest image for the exploration. How the suggestion workflow will be there is explained below.

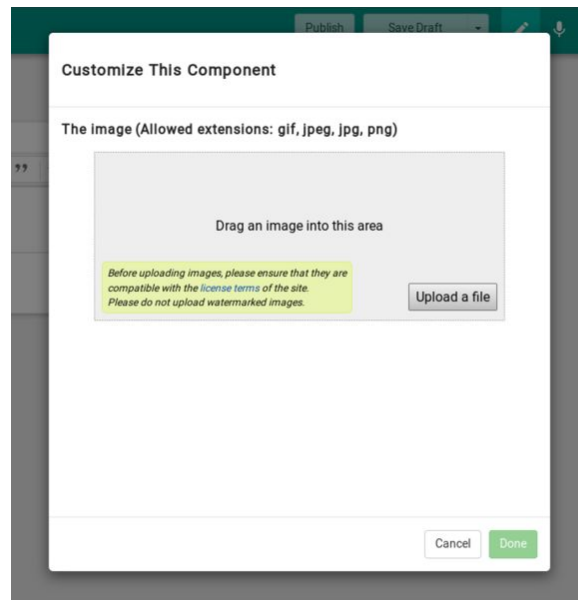
### 3. Image Suggestion Workflow

User control flow upon creating a suggestion



## Steps performed in suggesting image

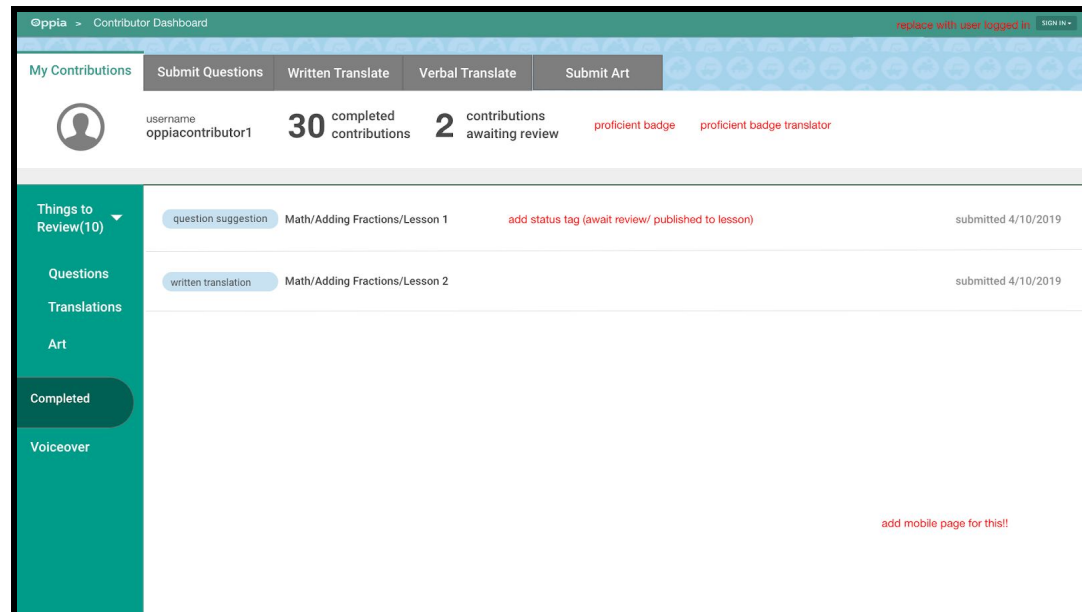
1. When the user clicks on the suggestion button, a modal for asking image will get opened.



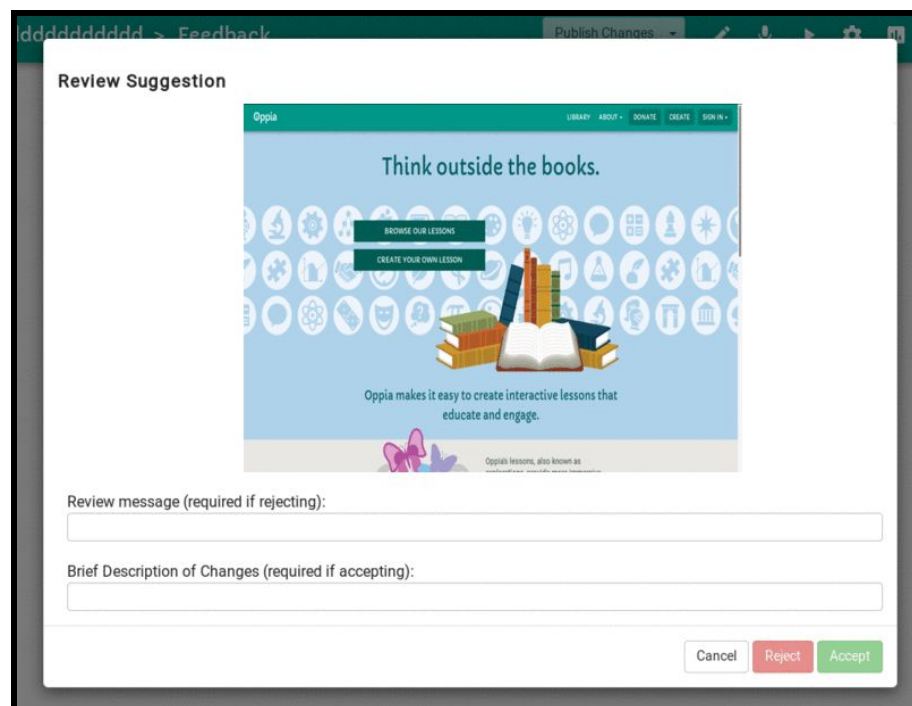
2. After uploading an image, a suggestion object will be created.

```
{
  "suggestion_type": "add_image_in_state_content",
  "target_type": "exploration",
  "target_id": "exploration_id",
  "target_version_at_submission": "exploration_version",
  "author_id": "<id_of_user_suggesting_image>",
  "description": "<description_of_suggestion>",
  "change": {
    "change_cmd": "upload_image",
    "state_name": "<state_name>",
    "image_info": {
      "id": "<image_id>",
      "src": "<image_src>",
      "author_id": "<author_id>"
    }
  },
  "placeholder_info": {
    "Placeholder_id": "id_of_placeholder",
  }
}
```

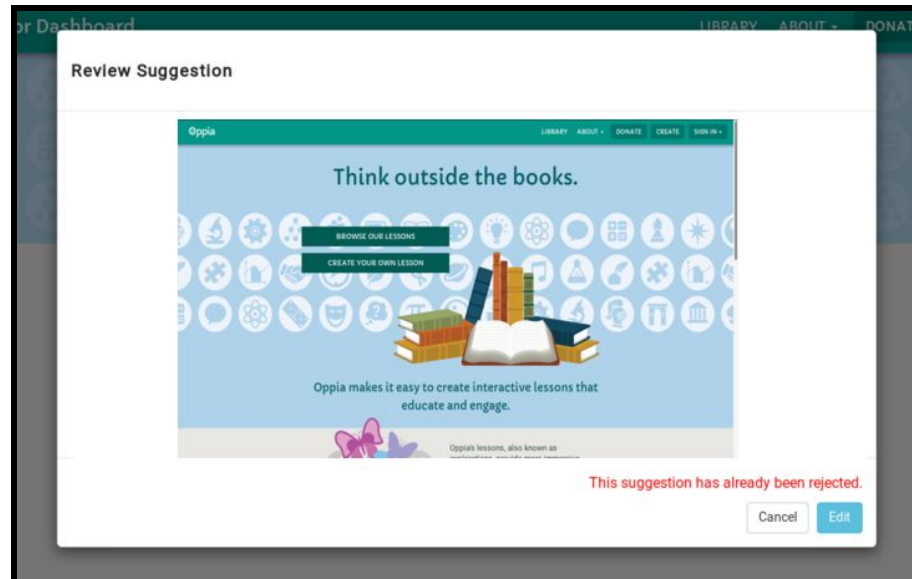
3. Now this suggestion object will be send to the server for creating a suggestion.
4. Now the suggestion will be show in the contributions dashboard page.



5. Now user who have permissions to review suggestion will review the suggestion.



6. If the suggestion get accepted, then the exploration card of the following state name will be replaced by the suggested exploration state content.
7. If the suggestion get rejected, then the rejected suggestion will be displayed on the community dashboard.



## Sequence flow

Files to be modified.

- **main.py:** New routes are to be created to create a new image request in the database as well as for viewing the image request. These are:
  - `/suggestionactionhandler/image_request/<target_id>/<suggestion_id>`: The class that this route calls would be a `controllers/image_request.py`. A new class called `SuggestionToImageRequestActionHandler` can be declared in `controllers/suggestion.py` which will call the required functions from `suggestion_domain.py` and `suggestion_services.py` in `core/domain` (which will also be created) to accept the submitted suggestion.
- **controllers/suggestion.py :**
  - Class `SuggestionHandler` (already created) : The already created suggestion handler class will be used to create a new type of suggestion.
  - Class `SuggestionToImageRequestActionHandler` : This class is used to handles operation when the suggestion get accepted by the reviewer.

Functions `accept_image_request_suggestion(suggestion_id, target_id)` will be called to accept suggestion.

- Class `ResubmitSuggestionHandler` (already created) : This class will be used to handle operation when the suggestion gets rejected.
- **domain/suggestion\_services.py** : This file has some following functions
  - `create_suggestion ( )` : This function is already created for creating a suggestion.
  - `accept_image_request_suggestion(suggestion_id, target_id)` : This function gets called in the class `SuggestionToImageRequestActionHandler` when the suggestion gets accepted. It updates the existing `exploration_state` content with the suggested html.
  - `Resubmit_rejected_suggestion ( )` : This function is already created for rejecting the submitted suggestion.
- **domain/suggestion\_registry.py** : This file has a class with functions in it
 

Class `SuggestionImageRequest ( )`

  - **`__init__( )`** : This function initializes the suggestion object of type `SuggestionAddImageInStateContent`. This suggestion object is of suggestion type `add_image_in_state_content`.
  - **`validate ( )`** : This function validates a suggestion object of type `SuggestionAddImageInStateContent`. It validates various things like check whether the change cmd is `replace_image_placeholder` or not, checks whether the change object is an instance of `ExplorationChange ( )` or not.
  - **`pre_accept_validate ( )`** : This function validates the change object before accepting by checking whether the state in change object exists in exploration or not. Because there may be a case after the suggestion has made the state has been deleted by the creators. Along with this there is also validation that the request of the placeholder exists in the state or not by checking the placeholder num.
  - **`get_change_list_for_accepting_suggestion ( )`** : This function returns a list containing change object of the suggestion.
  - **`accept ( )`** : This function gets called when the suggestion got accepted. It calls the functions like `get_change_list_for_accepting_suggestion ( )` and `update_exploration( )` for updating the exploration.
  - **`pre_update_validate ( )`** : This function performs the pre-update validation. This function needs to be called before updating the suggestion

Files To be modified for frontend implementation in folder

- core/domain/exp\_domain.py (add change cmd and property name)
- core/domain/exp\_services.py (edit function apply\_change\_list( ))
- core/domain/state\_domain.py (add functions for inserting suggested image in place of placeholder)
- core/templates/dev/head/pages/suggestion\_editor
  - SuggestionModalService.js
  - ShowSuggestionModalForCreatorView.js
  - ShowSuggestionModalForCreatorViewService.js
  - ShowSuggestionModalForEditorView.js
  - ShowSuggestionModalForEditorViewService.js
  - ShowSuggestionModalForLearnerLocalView.js
  - ShowSuggestionModalForLearnerLocalViewService.js
  - ShowSuggestionModalForLearnerView.js
  - ShowSuggestionModalForLearnerViewService.js
  - creator\_view\_suggestion\_modal\_directive.html
  - editor\_view\_suggestion\_modal\_directive.html
  - learner\_local\_view\_suggestion\_modal\_directive.html
  - Earner\_view\_suggestion\_modal\_directive.html

All files of this folder is need editing because a new type of suggestion will be created and this type of suggestion will need a new type of modal from creating to reviewing process.



## E2E testing plan

- **E2E test for inserting image is already implemented.**
- **E2E test for adding placeholder by creators/editors.**

### Steps to be done by creator

1. User logins
2. Clicks on create button to create an exploration
3. Add some info in exploration card.
4. Clicks in Image placeholder RTE component.
5. Create another card and add some info
6. Insert 2 image placeholders in same card.
7. End exploration and publish.
8. Loges out.

### Steps to be done by artist

1. User logins.
  2. User goes to profile dropdown menu and clicks on creator dashboard page.
  3. Go to assigned exploration.
  4. Click on upload button for uploading image in place of placeholder.
  5. Click on save button for saving exploration.
  6. Logged out.
- **E2E tests process for suggesting an image for the image request.**

### Steps to be done by creator

1. User logins
2. Clicks on create button to create an exploration
3. Add some info in exploration card.
4. Clicks on Image plugin.
5. Create another card and add some info
6. Insert 2 image placeholders in same card.
7. End exploration and publish.
8. Logged out.

### Steps to be done by user who suggests an image

1. User logins.
2. Redirects to community dashboard page.
3. Goes to Submit Art panel.
4. Click on the exploration.
5. Click on any of the placeholder present in the exploration.
6. Click on submit art button.

7. After clicking on submit art button, modals open showing the exploration card where the clicked placeholder present and open modal have a button for suggesting an image.
8. User clicks on suggestion button.
9. Modal opens and user uploads image, write a caption and brief description of the image.
10. Clicks submit button.

## Privacy

One question always come in mind before adding new feature in the product

**Does this feature collect new user data or change how user data is collected ?**

This feature does not collect new user data and also not change how user data is collected. The main data that's needed for the new feature is the exploration data. All things are to be done with help of exploration data like for adding images, deleting images. But suggestion process also not collect new user data and change user data.

## Security

This feature doesn't provide any new opportunities for users to gain unauthorized access to user data or otherwise impact other users' experience on the site in a negative way.

## Milestones

This project split into 3 milestones. Following things are to be done in respective milestones.

- **Milestone 1**

Images become independent from exploration content. Image source is no more gets saved in exploration content and creators/editors can add image placeholders in RTE content, and there is a function which provides the full list of unfilled image placeholders. Change command for adding images is created and unit tests for the newly created change command and functions are added.

- **Milestone 2**

New type of suggestion is created for suggesting an image for the exploration. UI for creating and review the suggestion got created. Unit tests and integration tests were added.

- **Milestone 3**

E2E tests for adding image placeholder added and e2e test for the suggestion workflow for the new type of suggestion created. Added documentation about image assets, placeholders and the suggestion workflow for suggesting an image.

## PR descriptions and their relevant dates

PR Description	Timeline	Dates of opening PRs	Expected Dates of merging PRs	No. of days taken before opening PR
PR for 1. Creating one more field in state model 2. Creating one more class ImageAssets 3. Creating new change command 'add_image' 4. Edit existing edit_state change command.	May 30 - June 5	June 5	June 8	7 days
PR for 1. Editing the image RTE component for adding the functionality of populating the images tags with the respective info 2. Creating and applying change object and command for image.	June 6 - June 10	June 10	June 13	5 days
PR for creating migration job for image assets	June 11 - June 15	June 18	June 20	5 days
PR for creating image placeholder and its change command	June 16 - June 21	June 21	June 24	6 days
Week for testing everything that has been implemented till yet.	June 22 - June 28 (Milestone 1 evaluation)			
PR for writing one off job for the image assets	June 29 - July 3	June 3	June 6	5 days
PR for creating a new type of suggestion called image request and add unit tests and integration test for new type of suggestion.	June 4 - July 10	July 10	July 13	7 days
PR for implementing UI modal for suggesting an image and add relevant unit tests.	July 11 - July 14	July 14	July 17	4 days
PR for implementing UI modal for seeing the suggested images in the contributions dashboard and add relevant unit tests.	July 15 - July 19	July 19	July 21	5 days

Week for testing everything that has been implemented till yet.	July 20 - July 26 (Milestone 2 evaluation)			
PR for reviewing the suggestion and add relevant unit tests.	July 27 - August 1	August 1	August 4	5 days
PR for adding e2e test for adding placeholder by creators/editors.	August 2 - August 8	August 8	August 11	7days
PR for adding e2e tests for testing contributions dashboard and suggesting an image for the image request.	August 9 - August 15	August 15	August 18	7 days
Week for testing everything that has been implemented till yet.	(August 16 - August 26)Milestone 3 Evaluation			

## Summer Plans

### Timezone

I would be in India throughout the duration of summer and hence would follow the Indian Standard Time (GMT +5:30).

### Time to commit for the project and other obligations

I have no commitments during summer. From 25th May my semester breaks start and thus, I would be able to put in at least 7-8 hours a day, which could be more if the project demands it. On weekends also, I'll put at least 4-5 hrs, which could also be more if the project requires it. Our semester starts on July 16, but it is not a heavy semester for us and in the beginning, not much work would be there also. Hence, during this time, I would be able to put 4 - 5 hours on weekdays, depending on the day and on the weekends, I could put more time (6-8 hours), to make up for any pending work.

From 15th August some semester works starts so my working hrs become 2-3 hrs less as compared to summer. And in weekends there is no decrease in working hrs.

## Communication

What is your contact information, and preferred method of communication?

My contact information:

**Mobile number:** +91-9643906878

**Email ID:** keshavbathla2017@gmail.com

**Github handle (gitter):** @import-keshav

Oppia is very active on gitter, so preferred method for most of the communication and meetings will be gitter. I will maintain daily devlogs as recommended by mentors to document the daily work.

How often, and through which channel(s), do you plan on communicating with your mentor?

We'll remain in touch over gitter(or Hangouts) twice a week to discuss the workflow to be followed or whenever I need advice.

## About Me

I'm Keshav Bathla, a second year undergraduate studying electronics and communication engineering at Jaypee Institute Of Technology, Noida(india) and an active member of Open Source Developers Club and Creative and Innovative Cell of Electronics IIIT Noida.-