

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.

Goals

1. We would like to make it possible to have dedicated artists for explorations: this will allow creators to assign artist roles, and allow the art development workflow to happen in parallel with other creative work.
2. 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.

Why do we need artist dashboard?

If we think deeply we should find that artist play an important role in the exploration creation process. See exploration creators can't be expert in every domain though he/she teaches well but maybe his/her drawing is not good and for that work, he/she need some other person who can do much better work then him/her. And we know 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. So to help creators and make explorations much better so that there should be good communication between exploration and learner, we need one dedicated artist which plays a vital role in creating explorations.

The above paragraph tells the importance of images but one question again comes in mind that why we still need artist dashboard and what are the benefits of it?

- **Large Collaboration.**

Currently, in exploration creation process only creators and some other users are only involved, but the ideas for improving exploration is just limited to few people.

Though suggestion feature helps to get ideas from other users to make it better, still it is used by only people that learn from this exploration but then what about other people who play exploration other than this?

Whenever a new exploration is created it is just known by only some few people until it's not getting famous. But artist dashboard gives functionality to creators to get ideas from all the people which helps in getting a large contribution to the exploration creation process.

- **Easy to contribute.**

As creators need an artist for adding images, but creators also want that's permission to edit exploration is also limited to few people. So artist dashboard helps in solving this problem. Artist dashboard helps creators to get images from artist though without giving edit permissions to artists. As giving permissions to edit exploration to all those people who contribute in exploration is not a good thing as it will increase prone to errors. Though artist doesn't have edit permissions still they can insert images easily with the help of artist dashboard.

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_assests ".
3. Image_assests object

```
"Image_assests" = {
  "content_id" = {
    "Image_id" = {
      "src" : "<image_src>",
      "caption" : "<image_caption>"
      "description" : "<placeholder_or_image_description>",
      "Is_reference" : "<is_image_uploaded_reference_image>"
    }
  }
}
```

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 content id?

To get to know in which content of the state is the image is linked to.

Process of adding image by creators or editors

After clicking on image RTE component an object will be created

```

"Image_id" = {
  "src": "<image_src>",
  "caption": "<image_caption>",
  "description": "<placeholder_or_image_description>",
  "ls_reference": "<is_image_uploaded_reference_image>"
}

```

And a change object will be created

```

"change" : {
  "change_cmd" : "add_image",
  "State_name" : "<state_name>"
  "content_id" : "<content_id>",
  "Image_id" = {
    "src" : "<image_src>",
    "caption" : "<image_caption>"
    "description" : "<placeholder_or_image_description>",
    "Is_reference" :
    "<is_image_uploaded_reference_image"
  }
}

```

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?

Before applying change command it is checked that did the image id in change object exist in exploration content or not. And if no then the change command will not be applied.

As soon as the user 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

- A user 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 with check that is the image id exist in the state content or not.
- Now the exploration will get saved

So this tells that there is no change in inserting or displaying 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 asset object from dictionary
 - Functions : Functions for uploading image.
- Along with this we have to edit class State for adding one more variable 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, 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)

One more thing to be done in this step is to write a one-off job for data migration.

Why I have to do this?

There is some editing of data models like while introducing one more field(i.e image_assest) 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 have to remove <oppia-non-interactive-image> tag from state content to all explorations and add new image object in the image assets

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 asset field.

Files to be modified

- feconf.py
- assets/constants.js
- core/domain/exp_domain.py
- core/domain/exp_domain_test.py
- core/tests/test_utils.py

2. Adding new role in exploration creation process called Artist and functionality for updating existing image.

Creator inserts image placeholders in the exploration to give hint to others collaborators that here we need, an image in exploration.

But now we introduce one more collaborator in exploration creation process called artist.

What kind of things that an artist can do or not ?

1. Artist can update existing images.
2. Artist can upload images in place of placeholder.
3. Artist can't delete, insert image (reason for not doing this will be discussed later).
4. Artist is not allowed to edit exploration content.
5. Artist can't add or remove any interactions.
6. Artist can't add or remove any translation.

Things to be done to implement this.

- Add one more option in select dropdown menu of Settings page in Roles section for assigning role.
- Introduce one new permission in exploration. For this I have to add one more field(artist_ids) in data models of exploration rights.

Why to do this ?

The main purpose of doing this is to involve an artist in the exploration creation process. The role of an artist is to just add an images in the exploration through image placeholder only. We want an artist to just involved in adding images to exploration, which can be done by adding one more permission to exploration i.e **Artist**.

Currently there exists 4 types of permissions

- Manager (can edit permissions)
- Collaborator (can make changes)
- Translator (can do audio translations)
- Playtester (can give feedback)

None of the existing permissions allowed to just insert image. So to make this possible we have to make one new permission.

- Introduce a new change command for updating images in place of existing images.
 - Update image command :

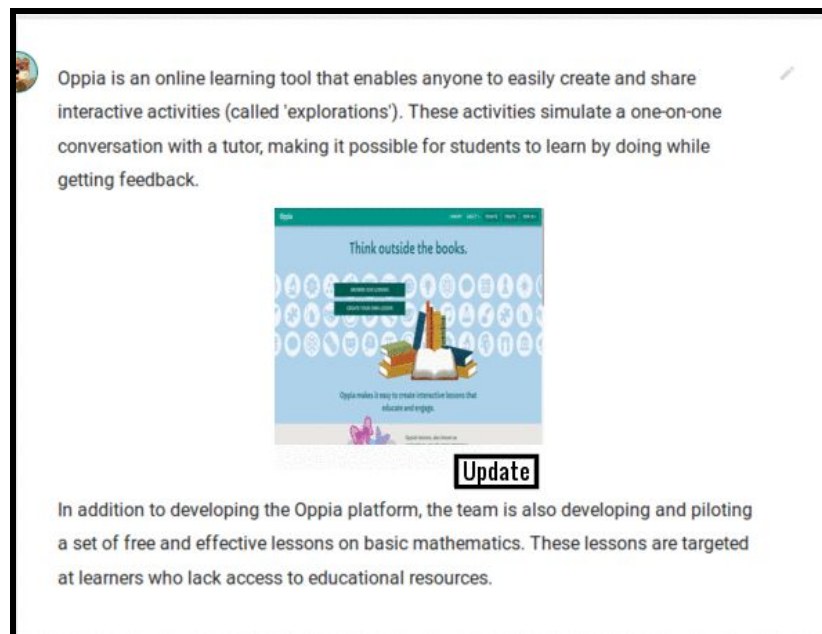
```

"change" : {
  "change_cmd" : "update_image",
  "State_name" : "<state_name>"
  "content_id" : "<content_id>",
  "image_info" : {
    "id" : "<image_id>"
    "src" : "<image_src>",
    "caption_of_image" : "<caption_of_image>",
    "description_of_image" : "<briefly_description>"
  }
}

```

How artist updates an image?

A button will be created that will be available below the image. So on clicking it a model will open for asking image info. And after clicks on save button the change object will be created.



Modal that opens for asking details

The image (Allowed extensions: gif, jpeg, jpg, png)

Drag an image into this area

Before uploading images, please ensure that they are compatible with the license terms of the site. Please do not upload watermarked images.

Upload a file

Caption for image (optional)

Briefly explain this image to a visually impaired learner

Description of Image (Example : George Handel, 18th century baroque composer)

Cancel Done

Now after some image actions we have to check the changes done by an artist are only related to images.

Why we have to check that there should only be changes related to images and How should we confirm that artist is just changing images in exploration ?

We want Artist to just involve in Art and graphics. So for this reason we have to check that there should only be changes related to images.

Whenever a change is done by the artist, then before saving an exploration it is checked that the change command in every change object of change list will be 'upload_image'.

After verification the function `update_exploration ()` will run to update exploration.

Why Artist can't delete, insert image?

Artist can delete and insert image but not now because adding this functionality needs more discussion and analysis of how to do this. We have to think this from various view like

- How to add image in the exploration at any place in the exploration where artist thinks there should be an image and without opening ckeditor also?
- How to allow artist to add image without affecting state content ?

- How to get which image did the artist wants to delete ?

There is one solution discussed below in alternate section to overcome this problem but still it should be implemented in separate project.

Files to be modified

For introducing a new field in exploration rights model

- `core/templates/dev/head/pages/exploration_editor/settings_tab/SettingTabs`
- `core/domain/rights_manager.py` (for adding one more field in rights object of exploration).
- `core/domain/rights_manager_test.py` (for adding test related to new permission called artist)
- `core/storage/exploration/gae_models.py` (for adding one more field in database for artist permissions).
- `core/storage/exploration/gae_models_test.py` (for adding tests related to adding of new field in exploration rights model)

Files to be modified for adding update image functionality

- `extensions/rich_text_components/Image/directives/ImageDirective.js`
- `extensions/rich_text_components/Image/protactor.js`
- `extensions/rich_text_components/Imagedirectives/image_directive.html`

Service used to implement this is **ExplorationRightsService** and function used is **saveRoleChanges()**.

For adding a new change command and function for performing image actions

- **`core/templates/dev/head/pages/exploration_editor/ChangeListService.js`** (for adding a new type of change object in exploration change list)
- **`core/domain/state_domain.py`** (for introducing a new functions for adding deleting and replacing image).
- **`core/domain/exp_domain.py`** (class `ExplorationChange` will be edit to introduce new change cmd)

One more thing to be done in this step is to write a one-off job for data migration.

Why I have to do this?

There is some editing of data models like while introducing one more permission(i.e artist) 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 by default permissions field should be an empty list.

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 artist permissions value to be empty list

Files to be modified

1. feconf.py
2. assets/constants.js
3. core/domain/exp_domain.py
4. core/domain/exp_domain_test.py
5. core/tests/test_utils.py

3. Add image placeholder feature.

What is image placeholder?

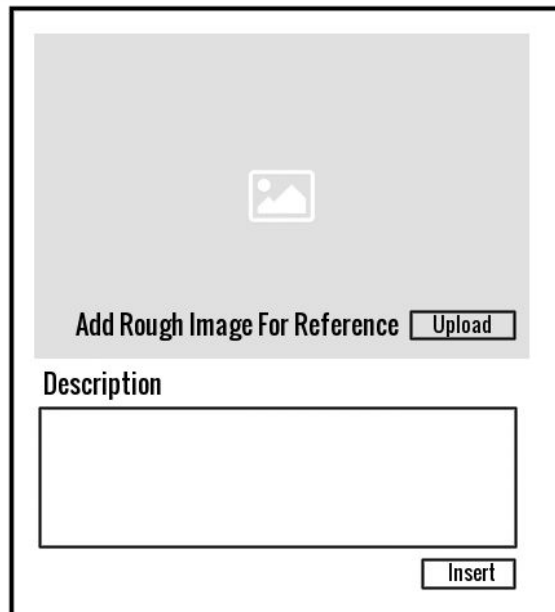
Image placeholder defines a space of an image in an exploration card that will be used by the creator to give a hint to an artist about an image should be inserted here. Artist is allowed to insert an image in exploration card through the image placeholder.

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 to add an image in exploration creator assign a job to some other user called an artist.
- His/Her (artist) job is to insert image in exploration, but while adding image how did he/she get where is the need of an image.
- So to overcome this creator adds image placeholder in exploration that defines space of an image in an exploration and gives a hint to an artist about an image should be inserted here.

Architecture of Image placeholder

Image placeholder template should be look like this in exploration card



The diagram illustrates the architecture of an image placeholder within an exploration card. It consists of a large rectangular container. Inside, there is a light gray rectangular area. Within this gray area, there is a small icon of a picture with a mountain and a sun. Below the icon, the text "Add Rough Image For Reference" is displayed, followed by a button labeled "Upload". Below the gray area, the word "Description" is written, followed by a large empty rectangular text box. At the bottom right of the container, there is a button labeled "Insert".

What is Add rough image for reference in above image?

Suppose creators might want to upload a rough image for reference which the artists can use as a reference. So to handle this we just add one option for inserting rough image which artist can take as an reference image.

How should be this implemented ?

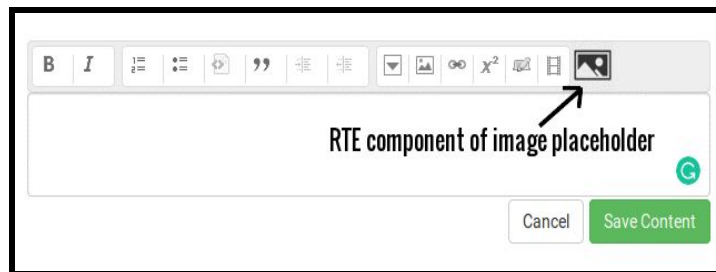
This can be implemented in 2 ways

- Make a another RTE component in CkEditor.
- Modify the existing image RTE component.

Case I

When we make another RTE component in CkEditor then the full process of how creator insert image placeholder and then how artist insert image in exploration will be something like this.

1. First creator clicks on image placeholder plugin for inserting image placeholder.



2. After clicking on image placeholder plugin a modal will be open for asking description about how an image should look like, to give an artist a hint. Its template will look like this.

 A screenshot of a modal form. At the top, there is a large gray rectangular area with a small image placeholder icon in the center. Below this area, the text 'Add Rough Image For Reference' is displayed, followed by an 'Upload' button. Underneath, the label 'Description' is shown above a large text input field. At the bottom right of the modal, there is an 'Insert' button.

After submitting details then the image object

(Placeholder without rough image)

```
"Image_id" = {
  "src" : "",
  "caption" : "<image_caption>"
  "description" : "<placeholder_or_image_description>",
  "Is_reference" : false
}
```

(Placeholder with rough image)


```
"Image_id" = {
  "src" : "<rough_image_src>",
  "caption" : "<image_caption>"
  "description" : "<placeholder_or_image_description>",
  "Is_reference" : true
}
```

will be created and the respective change command will gets created


```
"change" : {
  "change_cmd" : "upload_image",
  "State_name" : "<state_name>"
  "content_id" : "<content_id>",
  "Image_id" = {
    "src" : "<image_src>",
    "caption" : "<image_caption>"
    "description" : "<placeholder_or_image_description>",
    "Is_reference" :
      "<is_image_uploaded_reference_image>"
  }
}
```

3. After clicking on insert button exploration card will be look like this

Introduction




Oppia is an online learning tool that enables anyone to easily create and share interactive activities (called 'explorations'). These activities simulate a one-on-one conversation with a tutor, making it possible for students to learn by doing while getting feedback.



Description


Upload

In addition to developing the Oppia platform, the team is also developing and piloting a set of free and effective lessons on basic mathematics. These lessons are targeted at learners who lack access to educational resources.



CONTINUE →

- Now only users who have rights to edit exploration can upload an image. Upload button will be shown to them only. Now artist or other user clicks on upload button to upload an image and the final exploration card will be like this


Introduction



Oppia is an online learning tool that enables anyone to easily create and share interactive activities (called 'explorations'). These activities simulate a one-on-one conversation with a tutor, making it possible for students to learn by doing while getting feedback.



In addition to developing the Oppia platform, the team is also developing and piloting a set of free and effective lessons on basic mathematics. These lessons are targeted at learners who lack access to educational resources.


CONTINUE →

Now after clicking of upload button the respective change command will be created.

Upload image command : It is applied when artist uploads an image

```
"change" : {
  "change_cmd" : "upload_image",
  "State_name" : "<state_name>"
  "content_id" : "<content_id>",
  "image_info" : {
    "id" : "<image_id>"
    "src" : "<image_src>",
    "caption_of_image" : "<caption_of_image>",
    "description_of_image" : "<briefly_description>"
  }
  "placeholder_info" : {
    "Placeholder_id" : "id_of_placeholder",
  }
}
```

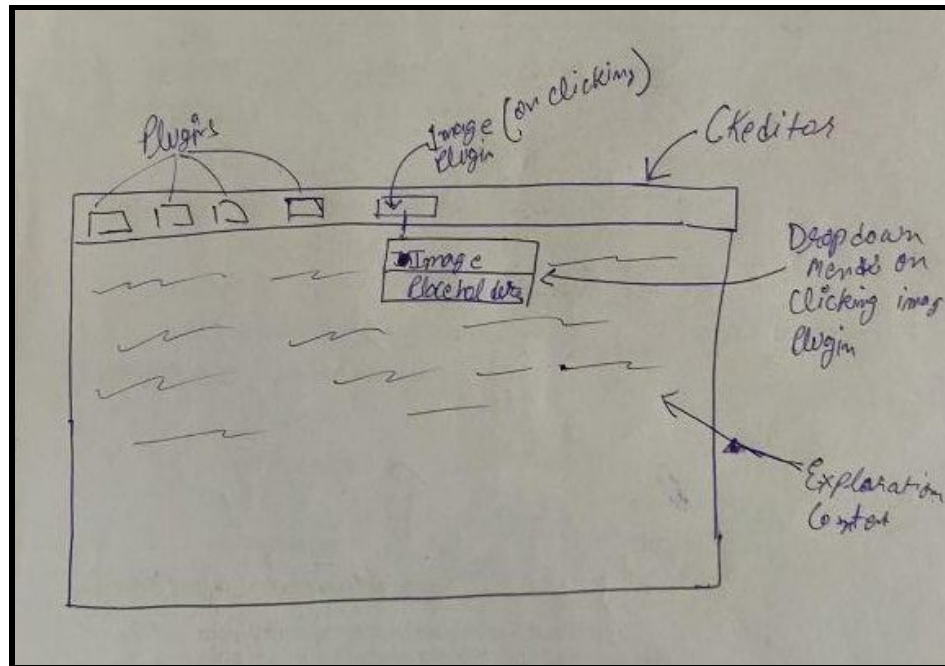
What if artist wants to remove uploaded image?

There will be button present under image where artist can click and remove the uploaded image.

Case II

Now when we implement image placeholder by modify the existing image RTE component the process will be similar to above but with some minor changes

1. First creator clicks on image plugin for inserting image or image-placeholder.
2. After this a small drop down box will open for asking whether creator want to add image or placeholder.



3. Then on choosing image placeholder option the respective image placeholder modal will open.

After adding placeholder info the image object (already mentioned above in case 1) will be created and add into the image assets of the respective content id.

So which is better ?

In my opinion second option is better due to following reasons

- Less code : Image placeholder plugin also contain the code of image plugin.
So it's better to use the existing code
- Less no. of plugins in the editor.

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.

Some points

- The function will check is there any object in content whose type is placeholder.
- There are 2 options for showing state names in the warning box
 - List view.
 - Simple text view (it means just a single text line having states names).

If we show the state names in list view then the list will become longer if the no of placeholders were large eg. no. of placeholders > 8 . And warning box will become very large which is not user-friendly.

I prefer to show state names in a simple text view which makes the warning box small and user-friendly.

Some general questions and its answers.

Did Artist can upload image by himself/herself in exploration content?

That's one of the important questions to be asked, so artist can't add image in exploration by himself/herself.

1. Why an artist can't add images? Because
currently there is only one way to insert an image in exploration and that is through image RTE, but it can be accessible to only those users who have permissions to edit exploration content. But the artist doesn't have permission to edit exploration content so he/she can't upload an image. If we still want this should be done then some change in structure needed which is discussed in artist panel of alternatives section.
2. What if an artist thinks that there is a need for an image in this exploration card?

If an artist thinks that there is a need for an image in this exploration card, then he/she can suggest that there will be an image, and suggestion will be made with the help of existing suggestion feature.

Why did artists didn't have edit access to all images and image placeholders in the exploration ?

Suppose an artist remove some placeholder or an image from the exploration where he/she thinks that there should not be a need of any image or placeholder, then after deleting of an image or placeholder there is no medium available where the creator can ask from artist why did you delete the particular image or placeholder

What if an artist adds and confirms wrong image and wants to change it again ?

In the case of Artist uploads the wrong image there is still an option to change it, but as soon as he/she publish the exploration then there is no option to change an image.

What If the creator doesn't like the image, did the placeholder will be inserted again?

Yes, but there is less chance of rejection of an image because we assume that the creator has some trust for the artist. That's why the creator has assigned him/her an artist role. But in case creator does not like the image then he/she will remove image and insert placeholder again with the better description.

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

The last step in this project is the image request view which is a view to allow people to suggest images in the exploration. But before moving further there is one question

How will we allow the community to see and suggest changes in private explorations?

As we all know the explorations which have placeholders were not present in the library because they were not the published exploration.

So how will community see the exploration?

To overcome this i will make one image_request flag. So what exactly will be happen that the explorations whose image_request flag is true and were unpublished will be shown in the common view.

If the exploration is unpublished and image_request flag is false were not shown in common view.

Published exploration with image request flag true is also not shown in the common view.

Now exploration which needs images are visible to community now

How will community suggest changes in private exploration ?

Community will suggest changes with the help of suggestion feature and how and what will be suggestion object will be described below in step 4

Where did the image_request flag will be stored?

I stored the image flag request in the exploration rights model.

4. Image Requests View.

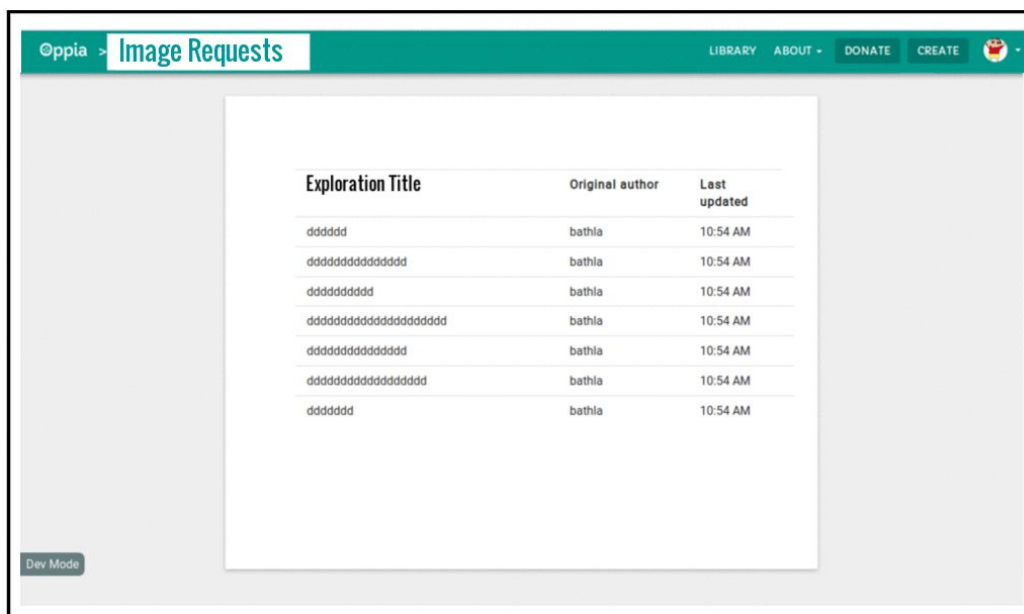
What's exactly is this?

The common view is a page where every user of oppia can suggest an image for the exploration. It most resembles like a stack overflow main page where there is a list of image requests will be shown and everyone is free to suggest an image without any restrictions. It's a way to organize the creation of graphics (using the existing suggestions framework) in a way that mimics the issue tracker on GitHub.

Why do we need this ?

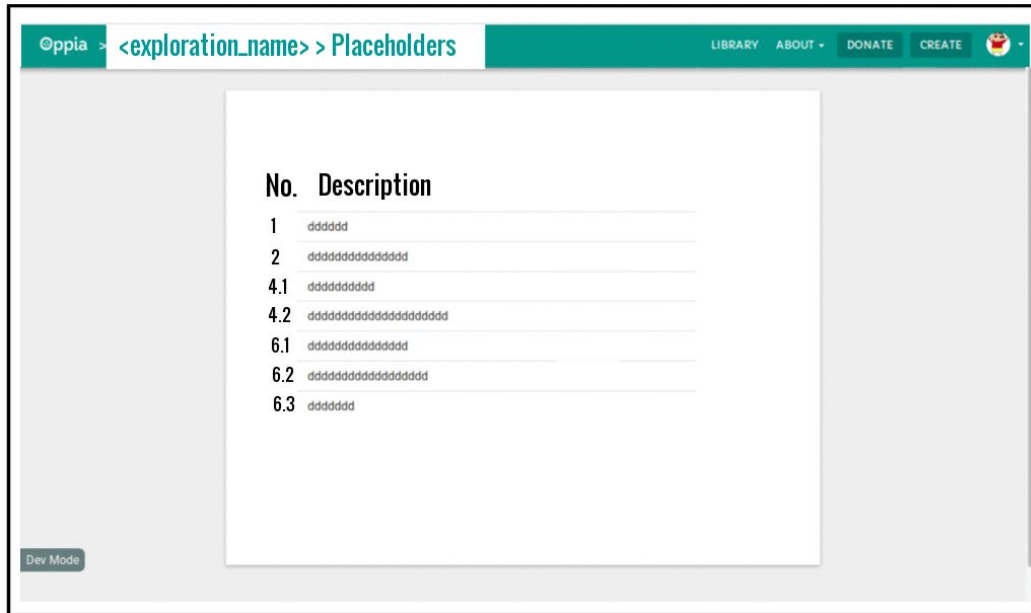
We don't only want an artist to insert images, but we also want that any other user can suggest an image (not insert like an artist) for the exploration that helps in increasing the collaboration of people in the exploration creation process. To implement this we implement a common view that displays all the explorations that want images.

Here is the UI of Image Requests view (Common view

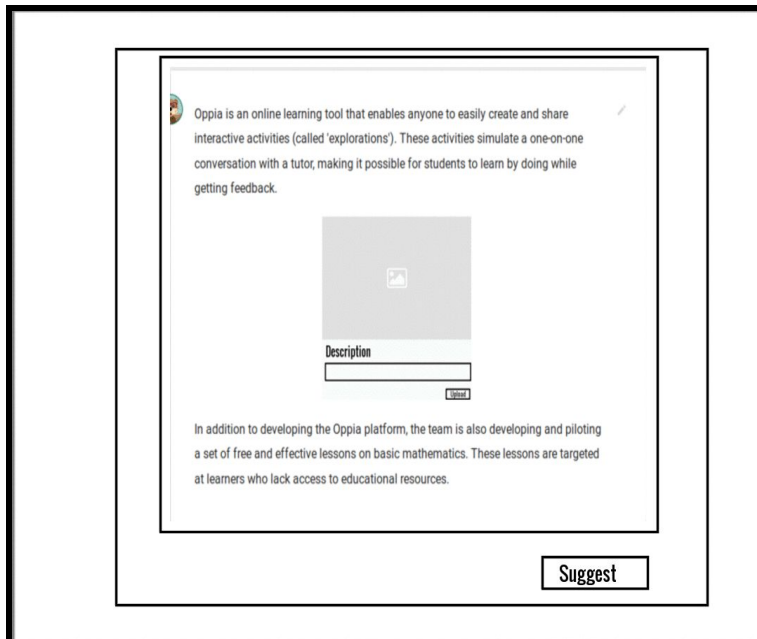


Now when the user clicks on any of the image requests, then a user will redirect to a page where the request of each placeholder will be shown.

Its UI Modal will be like this



Now when the user clicks on any of the rows, then a modal will open which shows the exploration card which helps the user to get an idea about which type of image is needed.



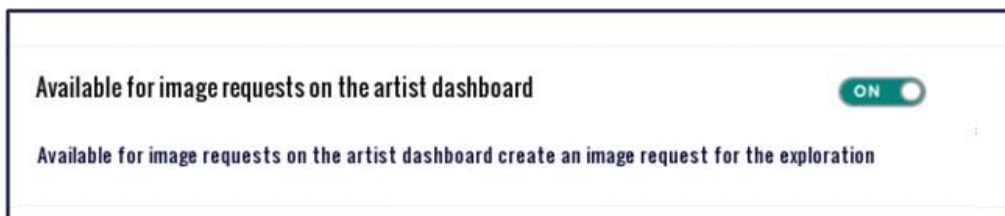
How creator add image request to common view ?

There is a button in the settings page, by clicking on it an image request will be created. This image request is for all placeholders in the exploration.

How to implement this ?

To implement this I add one more feature in settings page **available for image requests on the artist dashboard** whose value will be boolean.

Its a button like other feature Automatic Text-to-speech



Default it will be off, but when user turns it on then a flag in exploration card called **is_image_request_open** will becomes True.

Some questions and its answers

How will explorations will be displayed in common view?

When user redirects to this page a query will run in database that extracts all the explorations whose image_request flag is true and the status is unpublished. A json object containing all explorations id and their title will be rendered.

Json object is be like

```
"explorations" : {
  "exploration_id" : {
    "title" : "<exploration_title>",
    "author" : "<exploration_author>"
  }
}
```

How will I get the description of each placeholder along with linked exploration state content ?

When user clicks on any of the exploration then a json object containing list of placeholder objects present in the exploration will be come.

So the json object is be

```
"placeholders" = {
  "placeholder_id" = {
    "description": "<placeholder_description>",
    "State_name": "state_name_of_exploration"
    "content": "<content_in_which_placholdeder_present>",
  }
}
```

Here

Description is the detailed description of placeholder that will be shown in exploration state content in image placeholder.

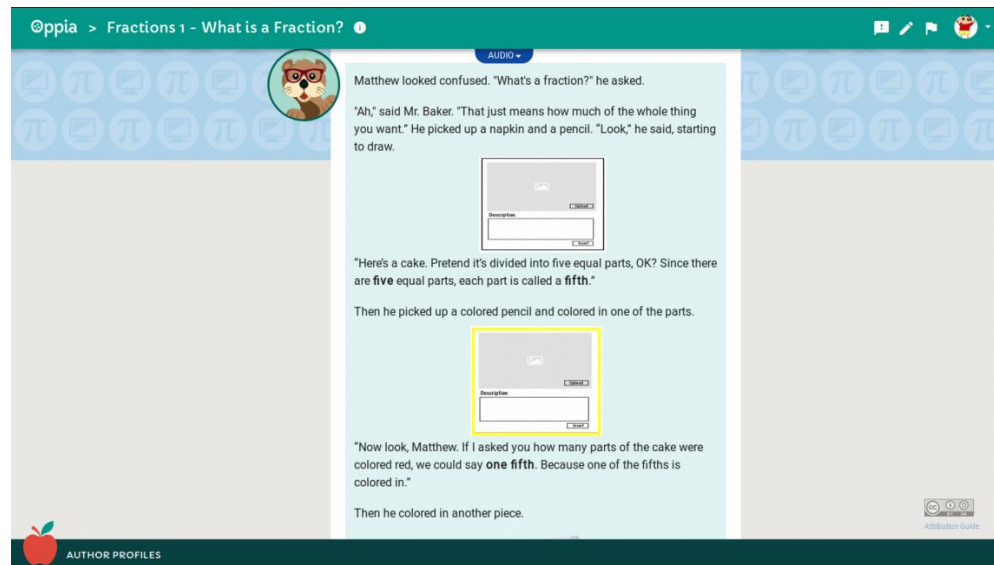
Content is the html of that content in which the placeholder is present. This content also contains <oppia-non-interactive-image> tag also.

What If a particular state content has more than 1 image placeholder, How did the requester know for which placeholder is this particular request?

A placeholder of the particular request will be highlighted in the exploration state content to get an idea about for which placeholder is this particular request.

Here's is the demo, so currently there are 2 placeholder in one card

Now for each placeholder, there is a different request, so when the user clicks on one of the requests then the modal opens for showing exploration content where placeholder will be highlighted to give hint about for which placeholder is that request.



How should i do this ?

The inline style will be added in the `<oppia-non-interactive-image>` of the exploration state content to highlighted the placeholder.

Suppose an image request is already opened, and editors insert a new placeholder, so how will creators add image request for the newly inserted placeholder ?

Every time the user redirects to the page image request placeholder list view, then each time the function will run for extracting placeholder and its info from the exploration data. So if image placeholder removed or added then automatically it will be not shown and shown respectively.

Suppose instead of inserting new image placeholder, editors have edited the existing exploration content though the image request has already opened. How will the new exploration content be shown in the opened placeholder request?

Its answer is same as the above questions, as user redirects to the page image request placeholder list view, then each time the function will run for extracting placeholder and its info from the exploration data.

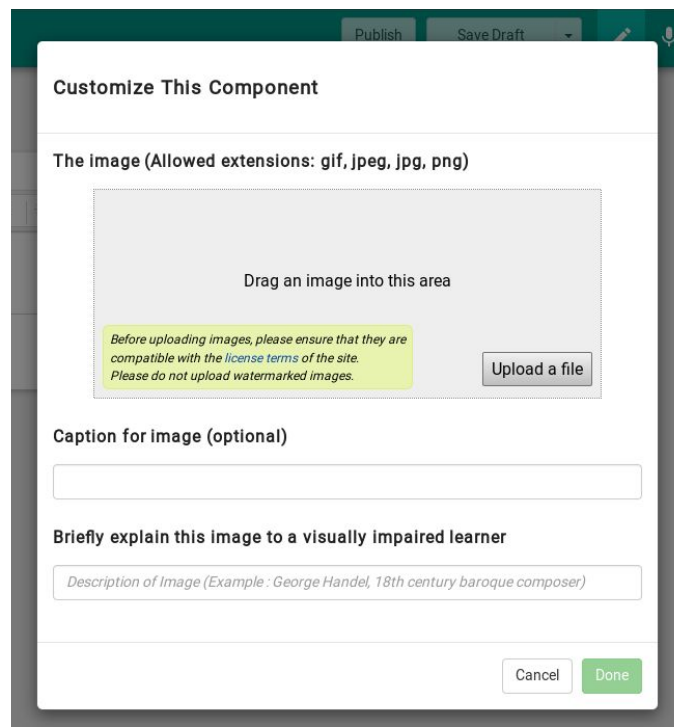
Flow of work in backend.

1. When the user comes to image request view.
 - Request come to redirect page image request view.
 - Extract all explorations from the database whose image request is open.
 - Send a response to the client.
2. When user redirects to image request placeholder list view.
 - Request come to redirect page image request placeholder list view.
 - Get exploration data whose exploration id is given.
 - A function will run to form the json object containing all placeholders info of the explorations.
 - Send response containing placeholders and its related info.

Till now, user is just viewing the image request but the next phase is the suggestion.

Steps performed in suggesting image

1. When the user clicks on the suggestion button, a modal opens for asking image and its related info. UI of modal for suggesting the image.



The image (Allowed extensions: gif, jpeg, jpg, png)

Drag an image into this area

Before uploading images, please ensure that they are compatible with the license terms of the site. Please do not upload watermarked images.

Upload a file

Caption for image (optional)

Briefly explain this image to a visually impaired learner

Description of Image (Example: George Handel, 18th century baroque composer)

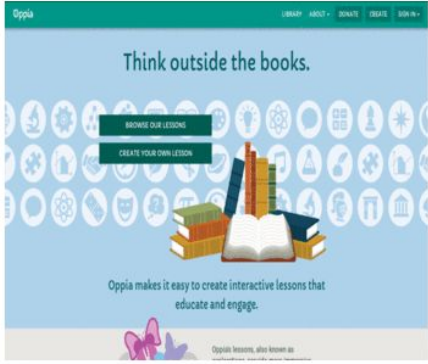
Cancel Done

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",
    "content_id": "<content_id>"
    "image_info": {
      "id": "<image_id>"
      "src": "<image_src>",
      "caption_of_image": "<caption_of_image>",
      "description_of_image": "<briefly_description>"
    }
    "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 feedback tab where the suggestion modal UI be like this.

Review Suggestion



Caption for image

oppia home page view.

Briefly description

Oppia is an online learning tool that enables anyone to easily create and share interactive activities (called 'explorations').


Review message (required if rejecting):

Brief Description of Changes (required if accepting):

Cancel Reject Accept

- Now user who have permissions to review suggestion will review the suggestion.
- If the suggestion get accepted, then the exploration card of the following state name will be replaced by the suggested exploration state content.
- If the suggestion get rejected, then the rejected suggestion will be displayed on the creator dashboard.

Review Suggestion



This suggestion has already been rejected.

Cancel Edit

Some questions and its answers ?

1. How will I insert suggested image in the exploration ?

A function will run that will remove the image placeholder object of the particular state with the respective suggested image.

How will i do it?

There is already implemented function `upload_image()` in `exp_services` for making an image object and remove the respective placeholder object.

2. How will I get to know for which placeholder is the suggested image ?

This can be done with the help of the state name. I know the state name so with the help of state name I extract exploration state content and replace the image placeholder tag with the suggested image.

What if there were more than 1 placeholder in one state ?

This can be resolved with the help of placeholder id store in suggestion object. A placeholder id will be unique of every placeholder. So if I get for which state is the suggested image then with the use of placeholder id I will get to know for which placeholder is the suggested image and replace the placeholder with the suggested image.

Code implementation is divided into 2 parts.

1. Backend / Server side code

2. Frontend / Client side code

● Backend / Server Side code

Implementation steps for creating an image request in the backend

- **feconf.py** : New constants have to be added. Some of which are:
 - `GET_IMAGE_REQUEST_URL` : `/image_requests`
 - `GET_IMAGE_REQUEST_PLACEHOLDERS_LIST_VIEW_URL` : `/image_requests/exploration/<exploration_id>`
 - `ACCEPT_IMAGE_REQUEST_SUGGESTION` : `/suggestionactionhandler/image_request/<target_id>/<suggestion_id>`
- **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:
 - `/image_requests` : The class that this route calls would be a part of `controllers/image_request.py`. A new class called `ShowImageRequests` can be declared in this file which will call the required functions from

- image_request_domain.py and image_request_services.py in core/domain (which will also be created) to show the open image requests.
- /image_requests/exploration/<exploration_id> : The class that this route calls would be a part of controllers/image_request.py. A new class called ShowImageRequestPlaceholdersList can be declared in this file which will call the required functions from image_request_domain.py and image_request_services.py in core/domain (which will also be created) to get placeholders of the exploration with the information like description of place holder.
 - /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/image_request.py** : In the classes defined below, the ones that are directly related to for showing an open image request or rendering image request page are defined
 - Class ShowImageRequests :Render the frontend html page when the url for showing the image requests is called. This will fetch the explorations whose image request is open and render them using the function self.render_template() .
 - Class ShowImageRequestPlaceholdersList : Render the frontend html page when the url for showing the image requests placeholder list view is called. This will fetch the placeholders from the exploration whose id is given and render them using the function self.render_template() .
 - **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 handels operation when the suggestion get rejected.
 - **domain/image_request_view_domain.py** : This file have some following functions

- `get_explorations ()` : Extract explorations from the database whose image request is open. This will return all explorations where `image_request = true`.
- `get_placeholders (exploration_id)` : Extract placeholders from the exploration data. This function will return a list of dictionary where dictionary have respective key value pairs.

```

"placeholders" = {
    "placeholder_id" = {
        "description" : "<placeholder_description>",
        "State_name" : "state_name_of_exploration"
        "content" : "<content_in_which_placholdeder_present>",
    }
}

```

- **domain/suggestion_services.py** : This file have some following functions
 - `create_suggestion ()` : This function is already created for creating a suggestion.
 - `accept_image_request_suggestion(suggestion_id, target_id)` : This functions gets called in the class `SuggestionToImageRequestActionHandler` when the suggestion get 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 having class with functions in it
 - Class `SuggestionAddImageInStateContent ()`
 - **`__init__()`** : This function initialize 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 exist 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 is exist 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

● Frontend / Client side code

In frontend i have to implement 3 things

1. A template for image request view.
2. A template for image request placeholder list view.
3. A template for modal that opens on requested placeholder.
4. Adding one more option in settings page.

Files to be Added

The folder `core/templates/dev/head/pages/image_request` is to be created in which the following files are to be added to make image request view.

- `image_request_common_view.html` : Template file for showing exploration whose image request were open.
- `ImageRequestCommonView.js` : Javascript file for making controller that handles operation in image request common view page.
- `image_request_placeholder_list_view.html` : Template file for showing placeholders of exploration in list view.
- `ImageRequestPlaceholderListView.js` : Javascript file for making controller that handles operation in image request common view page
- `exploration_state_content_of_placeholder_modal.html` : Template file for modal that shows exploration card where the clicked placeholder exist.
- `ExplorationStateContentOfPlaceholderModal.js` : Controller for handling operations on modal which shows exploration card.

Files To be modified 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.

Other things to be done other than this

1. Write a one-off job for data migration.

Why I have to do this?

In this step, there is a need for editing of data models to add a flag in exploration data to check whether an image request is open or not.

Adding one more field in the data model leads to create null value for the newly created field in older explorations data. And by default permissions field should be empty list and flag in exploration data should be false.

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 artist permissions value to be empty list and image request flag to be false.

Files to be modified

- feconf.py
- assets/constants.js
- core/domain/exp_domain.py

- core/domain/exp_domain_test.py
 - core/tests/test_utils.py
2. Add unit tests for the services and controllers created in files
 - controllers/image_request.py
 - controllers/suggestion.py
 - domain/image_request_view_domain.py
 - domain/suggestion_services.py

E2E testing plan

- **E2E test for inserting image is already implemented.**
- **E2E test for adding deleting images by artist.**

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. Clicks on the image request option present at top navigation bar.
3. After clicking on image request option user redirects to image request view.
4. Now user clicks on any of the rows of image requests thread.
5. After clicking on the row, user redirects to placeholders image request list view.
6. Now user clicks on any of the thread of placeholder list view.
7. After clicking on placeholder list view, 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.

Additional Considerations

There are some other things that should be considered while implementing this project

1. Introduce Artist Panel.

Why to implement this?

We want that artist to have full control over the art and graphics. To have full control there will be options to the artist for adding, deleting or uploading images and placeholders.

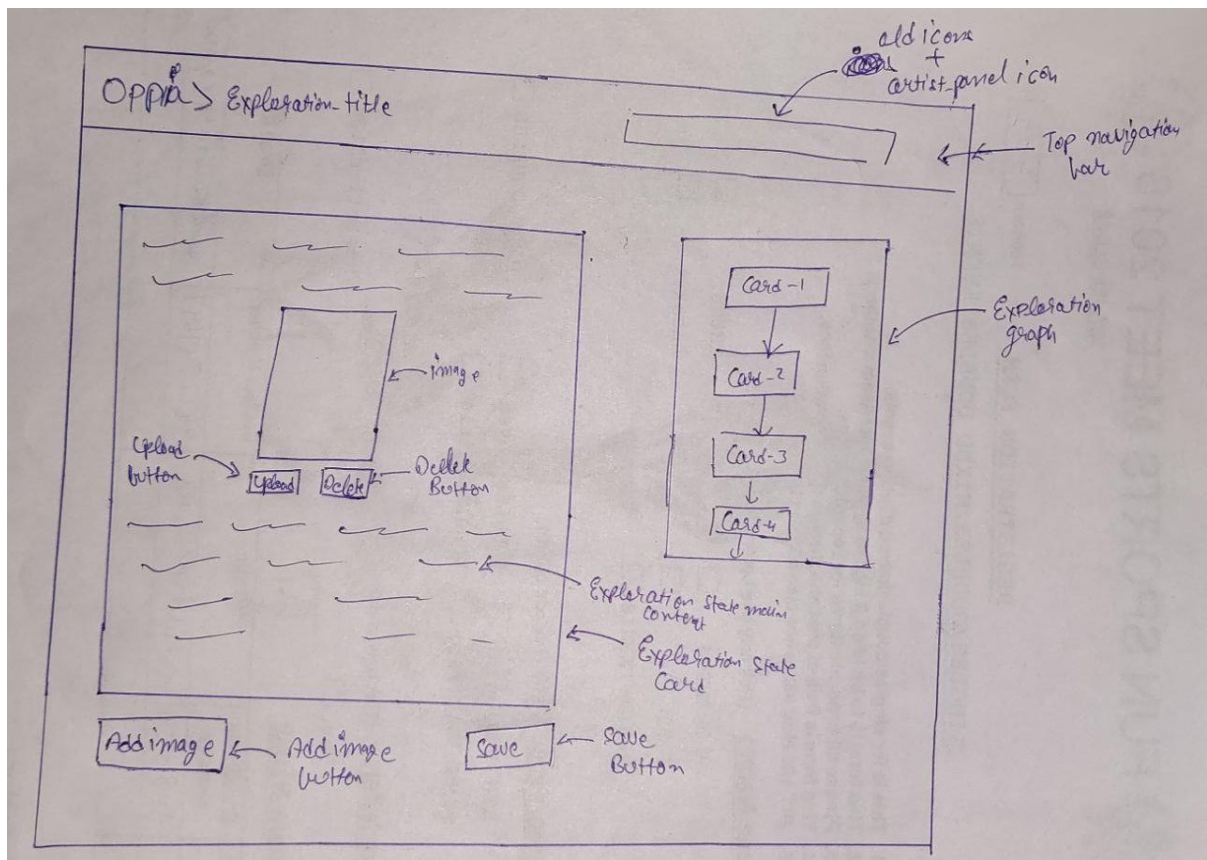
Before implementing this step there is a need of some change in structure. Currently when we add images then the image tags i.e `<oppia-noninteractive-image>` gets inserted in the state content. To make image independent from state content we have created an image assets field in state domain this allows artist to upload images in place of placeholder without affecting the state content.

Now to don't affect state content and allow artist to add image, then we don't have to store `<oppia-noninteractive-image>` in state content. Because if we store it, then for adding image we have to add `<oppia-noninteractive-image>` tag which effects state content so to overcome the problem we have to add images tag in state content only when we are going to display the exploration.

Now the point is how will we get where did the image tag to be inserted in state content while displaying exploration ?

The answer is Xpath. Xpath helps to know the exact location of image tag in state content.

This is how artist panel will look like.



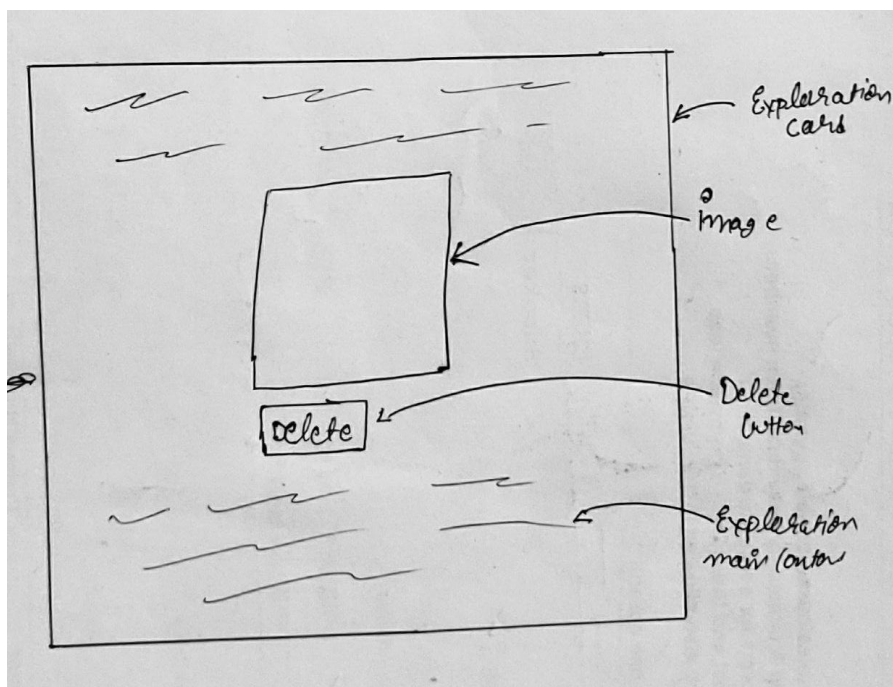
Things present in artist panel :

1. Exploration graph to show the overview of exploration to the artist.
2. Card that displays exploration state.
3. 5 buttons which are responsible for adding, uploading or deleting images and placeholders.

Card that displays exploration state.

Now before explaining function of each button one thing to say that there is some change in html of state content before displaying in the exploration card.

Exploration card view in artist panel will somewhat be like this



Here there is one delete button under the image to give functionality to delete the image if artist thinks there is no need of an image.

So basically a directive will be there which consists of `<oppia-non-interactive-image>` with some buttons. HTML of new directive

```
<div>
  <oppia-non-interactive-image></oppia-non-interactive-image>
  <button> Delete </button>
  <button> Upload </button>
</div>
```

So a directive called `<image-in-artist-panel>` containing `<oppia-non-interactive-image>` and buttons is append in state html, not `<oppia-non-interactive-image>` will be appended.

Buttons

Functions and working of each buttons present in artist panel.

1. Add image/placeholder :

It is used for adding an image. On clicking it a modal will open for asking some image details.

Things to be submit by artist

- Image
- Caption of image (optional)
- Description of image
- Line before image
- Line after image

What is line before and after image?

Since image can be inserted anywhere in the content maybe at start, in between or at the end. So artist have to decide in between which lines the image has to be inserted.

Why do we need this?

After entry of these 2 lines an xpath will be created.

Now we have all info regarding image now the change object will be created

```
"change": {
  "change_cmd": "add_image",
  "State_name": "<state_name_in_which_placeholder_present>"
  "image_info": {
    "content_id": "<content_id>"
    "id": "<image_id>"
    "xpath": "<xpath_of_image>"
    "src": "<image_src>",
    "caption_of_image": "<caption_of_image>",
    "description_of_image": "<briefly_description>"
  }
}
```

and will append in exploration change list.

2. Delete image/placeholder :

Deleting of an image is done by clicking on delete button present under the image or placeholder in the exploration card.

On clicking delete button then the respective change object will be appended in the change list.

```

"change": {
  "change_cmd": "delete_image",
  "state_name": "<state_name>"
  "image_info": {
    "content_id": "<content_id>"
    "id": "<image_id>"
  }
}

```

3. Upload image :

Uploading image will be done present under the placeholder.

On clicking placeholder button the respective change object will be created

```

"change": {
  "change_cmd": "upload_image",
  "State_name": "<state_name>"
  "content_id": "<content_id>",
  "image_info": {
    "id": "<image_id>"
    "xpath": "<xpath_of_image>"
    "src": "<image_src>",
    "caption_of_image": "<caption_of_image>",
    "description_of_image": "<briefly_description>"
  }
  "placeholder_info": {
    "Placeholder_id": "id_of_placeholder",
  }
}

```

and get appended in the change list.

4. Save : Save button to save the exploration.

Why i a not implementing in gsoc project ?

Due to some following reasons

1. I have to make images independent from state content which is must necessary step before implementing placeholders. So due to this i don't get time at gsoc for implementing this.
2. Implementing this needs some more discussion and analysis of how to insert images at random places which is one of the most trickiest part.

3. Adding this will create complexity like we have to add and remove images tag from state content while storing and displaying exploration respectively. It will leads to slow down of saving and displaying exploration.
So I think this should be implemented in separate project.

2. Allow creators to edit image request as well as placeholder description respectively.

There may be some cases where the creator adds an incorrect description, or there are some grammatical mistakes in the description. So to overcome this there will be a feature for editing the description.

3. Add labels for image request.

There should be some labels that should be displayed on the image request like **mathematics, python, data structures, graphs**, etc.

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.

Accessibility

1. For inserting image placeholder there is a plugin in RTE which helps creators to insert image placeholder in the exploration card.
2. For going to artist panel there is button present on top navigation bar in exploration editor page for redirecting to artist panel.
3. Image request view is accessible to all the users. For the landing, to an Image request page, there will be an option in the top navigation bar i.e image request where users can click to redirect Image request page.

4. For making an image request there is an option in settings page which helps editors to make image request for the exploration.
5. For users who want to suggest image for the exploration, there will be button on modal (that opens on clicking on any placeholder request) which opens an another modal for suggesting an image.

Milestones

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

- **Milestone 1**
 - Make images independent from stater content. Add unit tests for newly created functions responsible for making image objects and saving in state domain.
 - Write one of data migration job for newly created state object.
 - Creating a new permission called artist in exploration rights model.
 - Add unit tests for newly added permission
 - Write a one off data migration job related to permissions.
 - Create a new change command called `upload_image` for uploading image.
 - Add integration test for change command `upload_image`.
 - Writing functions on client side that add change to exploration change list on uploading image and checks is the change done by artist is only related to images.
 - Convert image RTE to plugin and placeholder option. All controller code will be written along with test.
- **Milestone 2**
 - Add flag `open_image_request` in exploration model and make client side code for opening image request.
 - Write one of data migration job for image request.
 - Implement functions for extracting exploration and placeholder from database and exploration state content respectively.
 - Add unit tests of functions or extracting exploration and placeholder from database and exploration state content respectively.
 - Implement controllers and url routing for displaying image requests and placeholders of explorations whose image request is open
 - Add unit test and integration test of function for image request view.
 - Add client side code for displaying explorations whose image request is open and their placeholders.
- **Milestone 3**
 - Creating a new type of suggestion called image request and adding service

- for accepting and rejecting new type of suggestion.
- Add unit tests and integration test for new type of suggestion called image request.
- Implement modals that will open on seeing the submitted suggestion.
- Write e2e test for the image request view and for suggesting an image.
- Fix all bugs that have been generated till now.
- Relevant documentation will be added describing the image request view, image placeholder RTE component and newly created permission i.e Artist.
- Buffer period for the final milestone. If all tasks are completed so far, an optional feature for editing the image placeholder description and labels to the image request will be created

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 <ol style="list-style-type: none"> 1. Creating one more field in state model 2. Creating one more class ImageAssests 3. Creating new change command 'add_image' 4. For editing existing test and creating new tests for the newly created functions. 	May 27 - May 31	May 31	June 3	5 days
PR for creating a new type of permission called artist in exploration rights model. Add unit tests for newly added permission.	June 1 - June 4	June 4	June 7	4 days
PR for editing the image RTE component for adding the functionality of populating the images tags with the respective info.	June 5 - June 7	June 7	June 10	3 days
PR for creating data migration job related to new state field called image assets	June 8 - June 11	June 11	June 14	4 days
PR for <ol style="list-style-type: none"> 1. Adding functionality for updating 	June 12 - June 16	June 16	June 19	5 days

existing image. 2. Checking did the changes done by artist were only related to image actions 3. Add tests for the newly created functions				
PR for converting image RTE to plugin	June 17 - June 19	June 19	June 21	3 days
PR for creating image placeholder and its change command	June 20 - June 22	June 22	June 25	3 days
Week for testing everything that has been implemented till yet. Fix bugs in case generated.	June 23 - June 28 (Milestone 1 evaluation)			
PR for adding flag open_image_request in exploration model and make client side code for opening image request.	June 29 - July 1	July 1	July 4	3 days
PR for creating a new type of suggestion called image request and add unit tests and integration test for new type of suggestion.	July 2 - July 6	July 6	July 9	5 days
PR for implementing functions for extracting exploration whose image request is open and function for extracting placeholders of the particular exploration Add unit test of these functions.	July 7 - July 10	July 10	July 13	4 days
PR for writing migration job that is required when we do some editing in data schema while implementing image request flag	July 11 - July 13	July 13	July 16	3 days
PR for implementing controllers and url routing for displaying image requests and placeholders of explorations whose image request is open. Add unit test and integration test of function for image request view.	July 14 - July 16	July 16	July 19	3 days
PR for displaying all explorations whose image request is open. (it contains all client side code like html,css and js files)	July 17 - July 20	July 20	July 23	4 days
Week for testing everything that has been implemented till yet. Fix bugs in case	July 21 - July 26 (Milestone 2 evaluation)			

generated.				
Pr for displaying placeholders of particular exploration for which the image request is open (it contains all client side code like html,css and js files)	July 27 - July 30	July 30	August 2	4 days
PR for implementing UI modals that will open on seeing the submitted suggestion by the creator in creator dashboard page.	August 3 - August 5	August 5	August 8	3 days
PR for implementing UI modals that will be open in feedback tab by the reviewer	August 6 - August 7	August 7	August 9	2 days
PR for adding e2e tests for image placeholder and image uploaded by artist.	August 8 - August 12	August 12	August 15	5 days
PR for adding e2e tests for testing image request view and suggesting an image for the image request.	August 13 - August 17	August 17	August 20	5 days
Week for testing everything that has been implemented till yet and Fix bugs in case generated. And Implement additional considerations in case all things going well and there is no bugs are left and generated	(August 18 - 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 start 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.-