

About You

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

One of the most important reasons for choosing Oppia is the work they are willing to do and that is, providing free education to everyone and myself being a teacher for kids. I know it feels great to teach students and right now they really need platforms like oppia to learn better. Secondly I've chosen this project that is oppia-android because of the code quality, architecture and using best practices available to build apps. It's also one of cleanest code I've seen for an android open source project.

Prior experience

I've been doing android development for about 3 years now and for more than 1.5 years I've been doing that in Kotlin with all the latest technologies which includes Dependency Injection, Jetpack Components, Coroutines etc. I have also been a runner up at an open source competition known as [BOSS](#) which is somewhat similar to gsoc and most of the work is open sourced. I have taken up workshops/sessions on git/github and basic android development in a few colleges and I also write [blogs on android development](#) and have published a few of them on Medium as well. Also i've published few apps one of the apps is

<https://play.google.com/store/apps/details?id=com.learnacad.learnacad&hl=en>

Which was made by entirely using firebase. Other than this there is open source project which is being maintained by me. Below is the link

<https://github.com/coding-blocks/CBOnlineApp>

Links to my Pull Requests

<https://github.com/oppia/oppia-android/pull/801>

<https://github.com/oppia/oppia-android/pull/723>

<https://github.com/oppia/oppia-android/pull/815>

<https://github.com/oppia/oppia-android/pull/836>

<https://github.com/oppia/oppia-android/pull/708>

Contact info and timezone(s)

Email - aggarwalpulkit569@gmail.com

Mobile - 9582054664

Preferred Mode of Communication - Hangouts for Essential Features rest can be explained on mail

Time Zone - My current timezone and time zone for the entire gsoc period will be India Standard Time

Time commitment

I'll be able to devote 35-40 Hours per week.

Essential Prerequisites

Answer the following questions:

- I am able to run a single android app test on my device



The screenshot shows the 'Test Results' tab in Android Studio. The top bar indicates 'Tests passed: 1 of 1 test - 4s 367ms'. The main area shows the test execution log for 'testHomeActivity_c...' on a 'Xiaomi Redmi Note 7S'. The log includes the command used to launch the test, the package name, and the test class. The status 'Tests passed: 1 of 1 test - 4s 367ms' is displayed at the top of the results pane.

```
Testing started at 10:49 AM ...

03/24 10:49:47: Launching 'testHomeActivity_c...' on Xiaomi Redmi Note 7S.
Running tests

$ adb shell CLASSPATH=$(pm path androidx.test.services) app_process / androidx.test.services.shellexecutor.ShellMain am instrument -r -w -e
targetInstrumentation org.oppia.app.test/androidx.test.runner.AndroidJUnitRunner -e clearPackageData true --no-window-animation -e debug
false -e class 'org.oppia.app.home
.HomeActivityTest#testHomeActivity_clickNavigationDrawerHamburger_clickOnHeader_opensProfileProgressActivity' androidx.test
.orchestrator/androidx.test.orchestrator.AndroidTestOrchestrator
Connected to process 7173 on device 'xiaomi-redmi_note_7s-2d80c4c7'.

Started running tests

Tests ran to completion.
```

Other summer obligations

I may be having my exams at the end of May. I'm not sure about that because of COVID-19 but for the rest of time period I will be obligated to organisation work.

Communication channels

Gitter and Github are more frequently used although sometimes hangouts as well.

Project Details

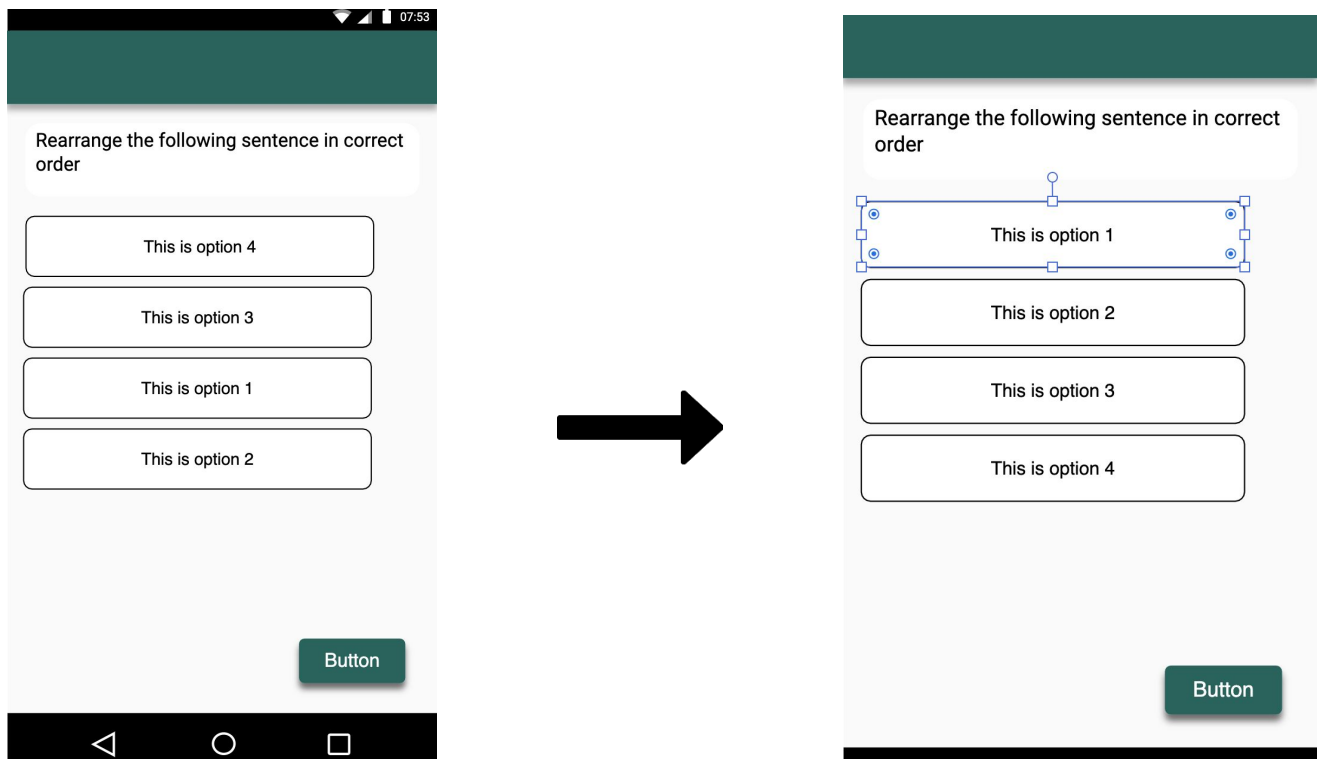
Product Design

In this project I'll be adding 3 types of interactions namely Drag and Drop, Ratios and Image Region Selection which will be for users who will be learning through the android app. The Ratios interaction is currently not available on the web so I will be working on introducing that on the web also for creators.

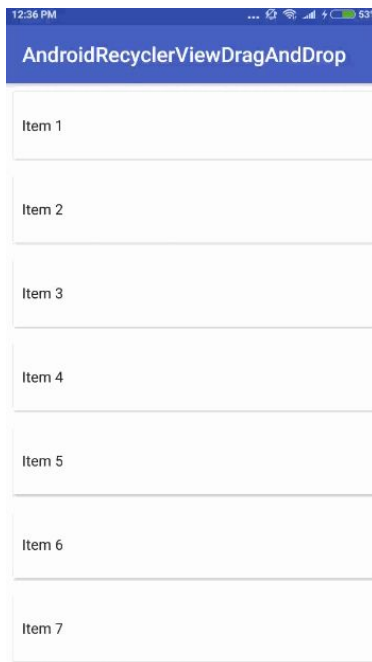
Following below each subsection shows one type of the interaction that one user may use

Drag drop and sort

In this type of interaction, the user will be given a list of options which may be shuffled in the order and the user has to rearrange the items in the correct option based upon the question description presented on the screen.



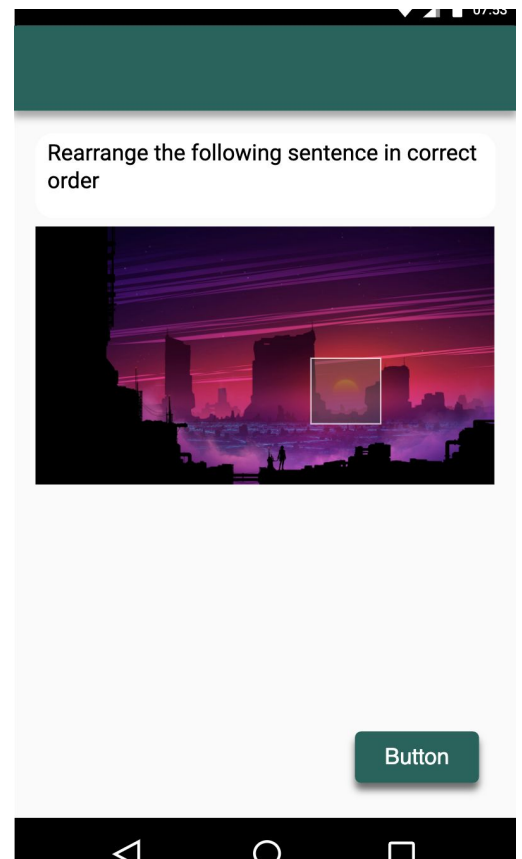
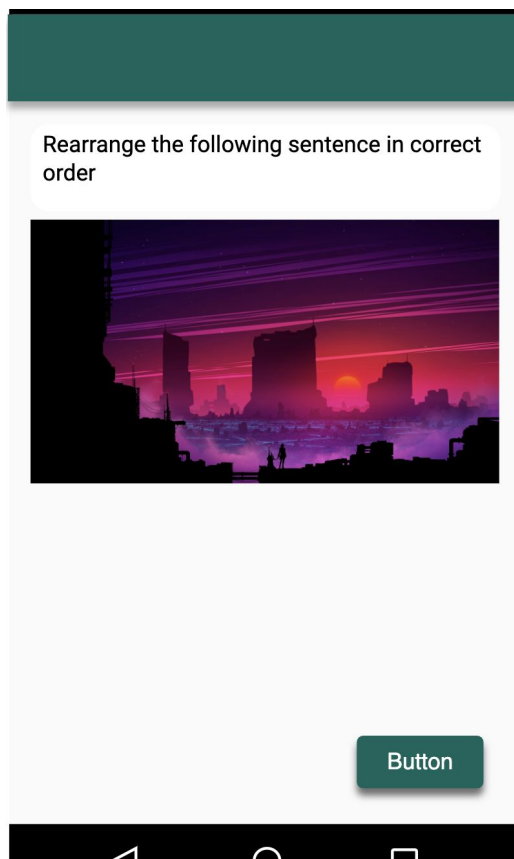
User has to hold an item and drag it to its correct position as shown on the right.

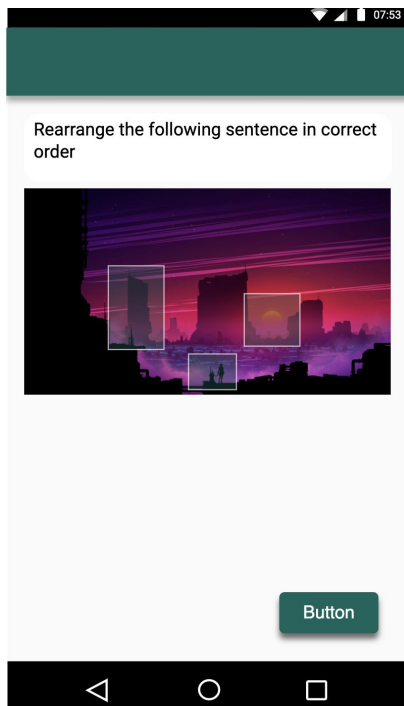


Once the user has rearranged a particular item, its new position will be saved and once the user is done with the correct ordering user taps proceed.

Image Region Selection

In this type of interaction the user will be presented with an image which can have multiple items/regions of interest which has to be selected by the user based upon the description of the question. Selecting anywhere other than the correct region will respond to an incorrect answer.

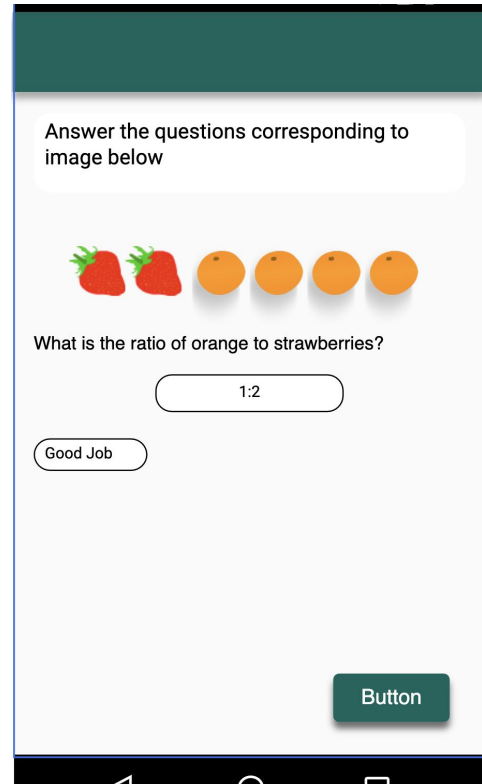
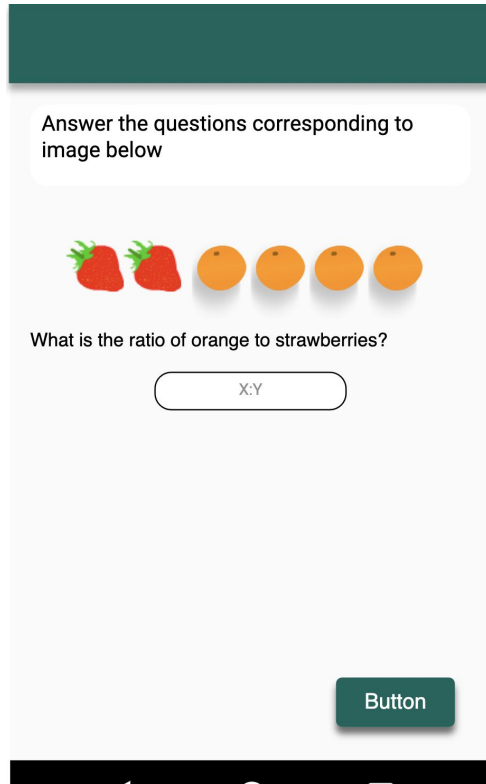




Hint Mode: If enabled by problem setter, the user can see all the possible regions which can be selected in image. It is intended to make it easier for the user to select a region.

Ratio Input Android

In this type of interaction the user will be presented with a question and 1 input box.



The user must submit solution to the question based on the constraints, finding the correct ratio

Web - Creator Dashboard

Creator's Journey

Upon choosing the RatioInput Interaction from the Math Tab the creator will be taken to the customization args where the creator has to give the hint text which will be exactly like Textinput but with only one row will be allowed also it would have a fixed format of writing $x : y$. This fix format is checked using a simple method that is to split the text around ':' and the size of the split array should be a minimum of two and after choosing this creator will be taken to Add Response Model. Feedback by anticipating various kinds of learner answers. The modal would look like this:

Add Response

If the learner's answer...

is equivalent to

1:2:3

Oppia tells the learner...

Nothing

And afterwards, directs the learner to...

(try again)

Cancel Save Response Save and Add Another

Rules: isEqualTo, IsEquivalentTo

In this interaction creators have to take care of the format of answers strictly as it will be something like $x : y : z \dots : n$. Finally, the creator would select the next interaction.

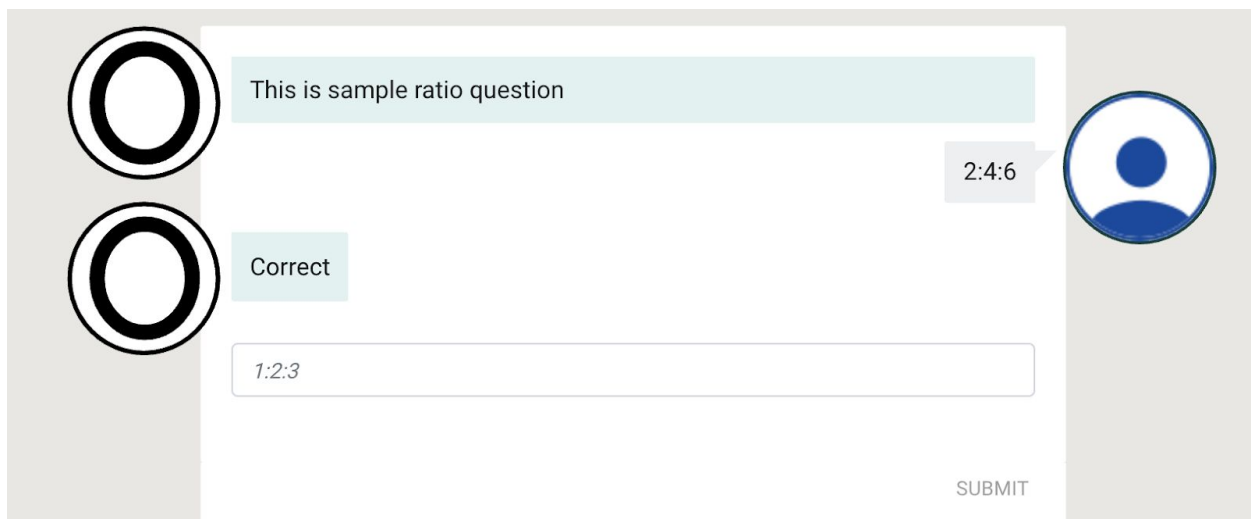
Also, the user input can only input values with the specific format with no decimal value allowing neither any signs +, -.

The creator can two choose either of the two rules which are explained below:

- **IsEqualTo:** This rule would expect one argument which should be a ratio and should be exactly equal to the value given by creator
- **IsEquivalentTo:** This rule would expect one argument which should be a ratio value but this will check for a common divisor for all the numbers present in the ratio object if they have a common divisor all the numbers will be divided by that common divisor to convert numbers into their lowest form and will be checked after that for the equality for both the numbers to the answer.

Web - Learner Journey

This is how an exploration that uses the Ratio Input Interaction would look like:



The screenshot shows a web interface for a ratio input interaction. On the left, there are two large circular icons, each containing a smaller circle. The top icon is white with a black border, and the bottom icon is black with a white border. In the center, there is a light blue rectangular box containing the text "This is sample ratio question". Below this box, there is a smaller light blue rectangular box containing the text "Correct". To the right of the "Correct" box, there is a white rectangular box containing the text "1:2:3". To the right of the "Correct" box, there is a small grey rectangular box containing the text "2:4:6". At the bottom right, there is a "SUBMIT" button. On the far right, there is a circular profile picture of a person with blue hair.

Rule : isEquivalentTo

The input for the learner would be simple numeric input and depending upon the rule which will have been chosen by the creator, the learner would be displayed with a warning if there is a mistake with the answer.

Technical Design

Architectural Overview

For adding different types of interaction process architectural processes would be similar. So I'll be explaining things in a generic format for the android app interaction but for the web codebase I'll be explaining things specific to ratio input interaction

Interactions for Android Client

- First, we need to define our modal and for Making Modal class we need to define our proto classes inside our model module which will be used for passing information through different layers of the app stack.
- Then we may require to define some classes inside our data module depending upon the changes which are brought in the backend
- For drag drop sort interaction we will be itemTouchHelper class which will be part of the utility module, similarly for image region selection interaction there will be various utility methods such getting coordinates of the image clicked, maintaining the aspect ratio of image etc which will be part of utility module
- We'll create custom view for ratio input and Image Region Selection which will be part of app module
- Once all the necessary classes are laid out then majority of the work will be the player package inside app module
- Then inside StateFragmentPresenter we need to register our interaction using viewDataBinder
- For registering any interaction, we'll have separate ViewModel which will be extending StateItemViewModel, as all the interaction are part of StateItem
- Then we need to add 3 classification inside domain module with necessary rules which will check answer is correct or not and then we need to declare these 3 types of classifier inside InteractionModule
- We also need to define ViewType inside StateViewModels

Ratio Input Interaction for Web

To create a new interaction, a directory will have to be created under `extensions/interactions/` for the new interaction with the corresponding name, i.e. `RatioInput`.

The general structure of this directory would be as follows:

Ratio -

- Directives -
 - `ratio-interaction.directive.html`
- Static -
 - `Ratio.png`
- `Ratio.py`
- `Ratio.ts`
- `__init__.py`

Implementation Approach

Drag drop and sort

To implement drag and drop functionality in any recyclerview we need a simple class i.e [ItemTouchHelper](#). `ItemTouchHelper` is a powerful utility that takes care of everything you need to add both drag & drop *and* swipe-to-dismiss to your `RecyclerView`. It's a subclass of [RecyclerView.ItemDecoration](#). In order to use `ItemTouchHelper`, you'll create an [ItemTouchHelper.Callback](#). This is the interface which will allow us to handle the changes which will happen on item drag. The main callbacks that we must override to enable basic drag and drop are:

```
getMovementFlags(RecyclerView, ViewHolder)
onMove(RecyclerView, ViewHolder, ViewHolder)
```

First we need to set the `movementFlags` because `ItemTouchHelper` class allows items to be drag up and down along with swipe and left and right but we need only drag so our `getMovementFlag` would look like something like this:

```

Override fun getMovementFlags(recyclerView RecyclerView,
    viewHolder RecyclerView.ViewHolder ):Int {
    val dragFlags = ItemTouchHelper.UP | ItemTouchHelper.DOWN
    return makeMovementFlags(dragFlags, swipeFlags)
}

```

Once we have setup the flags then we need to handle our onMove function which will be called whenever any item is dragged which would look something like this:

```

Override fun onMove(recyclerView RecyclerView,
    viewHolder RecyclerView.ViewHolder, target):Boolean {
    swapItems(viewHolder.getAdapterPosition(),
        target.getAdapterPosition())
    return true
}

```

Here swapItems will be a our own defined function which would something like give below :

```

fun swapItems(fromPosition: Int, toPosition: Int) {
    if (fromPosition < toPosition) {
        for (i in fromPosition..toPosition - 1) {
            options.set(i, options.set(i+1, options.get(i)));
        }
    } else {
        for (i in fromPosition..toPosition + 1) {
            options.set(i, options.set(i-1, options.get(i)));
        }
    }
    notifyItemMoved(fromPosition, toPosition)
}

```

Whenever user changes the order of any item a function will be called with position that is the old position and new position and then using these position we will update the original list as well as update the order of item in adapter using notifyItemMoved().Here options is a array list of options which will be generated from the API Response.After we just need to attach the ItemTouchHelper class with the recyclerview which can be done by Custom Adapter for RecyclerView to work with DataBinding

Image Region Selection

In this the first and the foremost thing required would be to maintain the aspect ratio of images which can be easily achieved by giving Glide the respective height and width which can be taken from imagePath from the api response. Below is the sample code to resize a image:

```
GlideApp
.with(context)
.load(imageUrl)
.override(400,200)
.into(imageView)
```

Considering that we are able to maintain the aspect ratio then we will be attaching our imageView to a custom class called ClickableImageArea which will be defining inside utility using [PhotoViewAttacher](#) and this library will be a foundation for this entire process

<https://github.com/LukasLechnerDev/ClickableAreasImages>

Let's take a look at ClickableImageArea class and its functions

```
class ClickableAreasImage(
    attacher: PhotoViewAttacher,
    listener: OnClickableAreaClickedListener
) : PhotoViewAttacher.OnPhotoTapListener {
    private val attacher: PhotoViewAttacher
    private var listener: OnClickableAreaClickedListener? = null
    var clickableAreas: List<ClickableArea>? = null
    private var imageWidthInPx = 0
    private var imageHeightInPx = 0

    private fun getImageDimensions(imageView: ImageView) {
        //To get Image Dimensions
    }

    fun onPhotoTap(view: View?, x: Float, y: Float){
        //Handle Tap of Image
    }
}
```

Once we have created a object of above class and passed a reference of the image view then we can define out clickable areas using a custom class ClickableArea which is defined below:

```
class ClickableArea {
    private val x1 = 0
    private val x2 = 0
    private val y1 = 0
    private val y2 = 0}
```

And using above class we add clickable areas which we get in API response to our ClickableAreasImage class object which we declared earlier sample code is given below:

```
val clickableAreas = arrayListOf<ClickableArea>()
// parameter values (pixels): (x1 coordinate, x2 coordinate, x1
coordinate, x2 coordinate,) and assign an object to it of available
clickableAreas.add(ClickableArea(500, 200, 125, 200,)
clickableAreas.add(ClickableArea(600, 440, 130, 160,String("Sun"))
// Set your clickable areas to the image
clickableAreasImage.setClickableAreas(clickableAreas)
```

Once this clickable areas are set then we just need to handle click inside our photoTap() function and there we can handle clicks of any image

Ratio Input Interaction - Android

The ratio input interaction will be very similar to the number input interaction as it will be having a single field only but in this case it will be custom view that is RatioInputView which will be having inbuilt validations for taking input as Ratio format only i.e x:y:z...n.The validation function would look like something give below:

```
fun isRatio(ratio: String): Boolean {
    val items = ratio.split(':')
    return if (items.size < 2) {
        false
    } else {
        items.stream().allMatch { it.toIntOrNull() != null }
    }
}
```

One Important factor would be that whether if the creator has allowed ratio values to be applicable with common divisor i.e. if user is entering two number which have a common divisor and when these numbers are divided by that common divisor gives their lowest term.For the above case we need to write classification method which will be having array as its argument and will be return the common divisor.If there is no common divisor 1 will be returned else the number.If classification

works fine array of numbers will be converted into their lowest term. This helper function would look something like given below :

```
// Function to return gcd of a and b
fun gcd(a: Int, b: Int): Int {
    return if (a == 0) b else gcd(b % a, a)
}
// Function to find gcd of array of numbers
fun findGCD(arr: IntArray, n: Int): Int {
    var result = arr[0]
    for (i in 1 until n) {
        result = gcd(arr[i], result)
        if (result == 1) {
            return 1
        }
    }
    return result
}
```

Let's take an example to understand this. Let's say we have input as 2:4:6:8 then findGCD function will return 2 as 2 is common divisor for all of them but if we have input as 2:3:6:9 then it will return 1 that means we cannot break down and result be sent as the original input but for first case the number will be divided by the common divisor then it would be sent as a result.

Ratio Input Interaction - Backend

The first step would be to introduce a new data type, *RatioExpression* that this interaction will require. The classes for these objects would be added in the `objects.py` file under `extensions/objects/models/`. They would look something like this:

```
Class RatioExpression(BaseObject):

Description: 'A string for ratio expression.'
default_value = '1:1'

SCHEMA = {
    'type': 'unicode',
    'validators': [{
        'Id': 'is_valid_ratio'
    }]
}
```

```
}
```

The validator functions for these classes will be added as static methods inside the Validators class present in `schema_utils.py`. `is_valid_ratio`: This method will return true if the expression:

- Starts with a number and ends with a number.
- Minimum number of elements must be 2.
- Between every two numbers there should be exactly one ‘.’.
- Does not contain operators (+, −, /, *, ^, .).
- Does not contain string or special characters.

The next step would involve creating the `AlgebraicExpressionInput` directory under `extensions/interactions/` which will contain the following files that would define the behavior of the interaction.

`RatioInputExpressionInput.py`: This will be the file for defining the interaction with all the basic things like name, description etc.

```
Class RatioInput(base.BaseInteraction):
```

```
    name = 'Ratio Input'
    description = 'Allow learner enter number in ratio format'
    display_mode = base.DISPLAY_MODE_INLINE
    is_trainable = false
    _dependency_ids = []
    answer_type = 'RatioExpression'
    can_have_solution = true
    show_generic_submit_button = true

    _customization_arg_specs = [{
        'name': 'placeholder',
        'description': 'Placeholder text(optional),
        'schema': {
        'type': 'unicode',
```

```
},  
'default_value': ''  
}]
```

Next will be defining our views inside directives which will be nothing but simple html files along with some rules and validations. Let's take a look at some validation that would be implemented:

- containsOnlyColonAndNumber : It will check the input contains only colon and numbers nothing else
- imbalanceRatioSequence : It will check that if the number of values are n then there should be exactly n-1 colon between each of them.

After that we will be needing to define our Rule whose implementation is given below:

- isEquivalentTo : The following steps will have to be carried out for this rule:
 - Split the expression with ' : ' as delimiters using the a simple split function
 - Now we have list of Numbers: we will try find a common divisor of these numbers using a function getLowestTerm()
 - Now if the above step is successful then we will be having two lists that creatorTerms and learner Terms. Iterating through the learnerTerms list we try to match it with one of the creatorTerms. If all learnerTerms match with a corresponding creatorTerm, this rule returns true.

Testing Approach

Ratio Input Interaction - Android

For ratio input interaction we will be needing to add one more view inside InputInteractionViewTestActivity. Then we will be adding method inside InputInteractionViewTestActivityTest like numbers button continue/submit button is disabled if field is not filled. Some more test like converting numbers to their lowest term, numbers are in the range of integers.

Drag Drop Sort Interaction

For this interaction we will be creating a new TestingActivity with a recyclerview which is attached with itemTouchHelper and sample list. There will be a corresponding TestActivity for the previously created activity. The test will include some simple test that whether list is displaying correct no of items, text is being displayed correctly or not and will contain some test for checking when item dragged from position x to y the values are saved in the list or not etc.

Image Selection Region Interaction

For this interaction we will be creating a new TestingActivity with our custom image view and a corresponding TestActivity for the previously created activity. There will be tests like whether image is being displayed or not, there are any selectable regions for the image or not, image aspect ratio is matches or not, handling different clickable regions etc.

Ratio Input Interaction - Backend

Ratio interaction includes two major services that need to be checked thoroughly i.e. rules and validation. Both of them will have a corresponding spec file for unit testing. These files will contain tests for each rule that the corresponding interaction supports.

Some of the tests that will be written for this interaction would be like checking if the interaction object return on submission is ratio only and not anything else, checking if interaction is correctly displayed etc.

Milestones

Milestone 1

Key Objective: Fully implement the drag-and-drop sorting interaction in the Android app.

No.	Description of PR	Prereq PR numbers	Target date for PR submission	Target date for PR to be merged
1.1	Adding new interaction objects for DragAndDrop	None	18.5.2020	19.5.2020
1.2	Add Interaction State in StateRetriever(Use ENUMS ?)	None	19.5.2020	20.5.2020
1.3	Adding Interaction Module and declare that in application Component	None	20.5.2020	22.5.2020
1.4	Add classifier hasElementXAtPositionY	1.3	21.5.2020	23.5.2020.
1.5	Add classifiers checkEquality	1.3	21.5.2020	23.5.2020
1.6	Add classifier checkEqualityWithIncorrectPosition	1.3	22.5.2020	24.5.2020
1.7	Add classifiers isEqualToOrdering	1.3	22.5.2020	24.5.2020
1.8	Add classifiers isEqualToOrderingWithOneItemAtIncorrect Position	1.3	23.5.2020	25.5.2020
1.9	Add classifiers hasElementXBeforeY	1.3	23.5.2020	25.5.2020
1.10	Map classifiers rules to DragDropModule using annotation class	1.4	24.5.2020	25.5.2020

1.11	Registering Interaction inside InteractionObjectTypeExtractorRepository	1.1	25.5.2020	26.5.2020
1.12	Add Helper class for DragDropSort - ItemTouchHelper class which will have on swiped callback method	None	26.5.2020	30.5.2020
1.13	Low-fi : Add Drag Drop Interaction - Add DragDropSort item View and its corresponding ViewModel	1.7	30.5.2020	4.6.2020
1.14	HighFi : Drag Drop Interaction	1.8	3.6.2020	6.6.2020
1.15	Adding accessibility for DragDropSort	None	10.6.2020	11.6.2020
1.16	Adding new interaction objects for ImageRegion	None	6.6.2020	7.6.2020
1.17	Add Interaction State in StateRetriever for ImageRegion	None	8.6.2020	9.6.2020

Milestone 2

Key Objective:

- Fully implement the image region interaction in the Android app.
- Implement backend support for ratio input.

No.	Description of PR	Prereq PR numbers	Target date for PR submission	Target date for PR to be merged
2.1	Adding Interaction Module and declare that in application Component	None	12.6.2020	13.6.2020

2.2	Add classifier isInRegion	2.1	13.6.2020	15.6.2020
2.3	Map classifiers rules to DragDropModule using annotation class	2.2	15.6.2020	15.6.2020
2.4	Registering Interaction inside InteractionObjectTypeExtractorRepository	2.3	16.6.2020	16.6.2020
2.5	Add Helper class for Selecting Image Region - PixelPosition & getPixelPosition for a particular image	None	20.6.2020	22.6.2020
2.6	Add PhotoView Dependency and function to maintain aspect ratio	None	22.6.2020	24.6.200
2.7	Add ClickableImageAreas Class	2.6	26.6.2020	30.6.2020
2.8	Low-fi : Add ImageRegion Selection Interaction - Add ImageRegion item View and its corresponding ViewModel	2.7	4.7.2020	7.6.2020
2.8	HighFi : ImageRegion Selection Interaction	2.8	7.6.2020	9.6.2020
2.9	Adding Ratio Input to Backend	None	15.7.2020	20.7.2020

Milestone 3

Key Objective: Fully implement the ratio input interaction in the Android app.

No.	Description of PR	Prereq PR numbers	Target date for PR submission	Target date for PR to be merged
3.1	Adding new interaction objects for Ratio Input	2.9	20.7.2020	21.7.2020

3.2	Add Interaction State in StateRetriever for ImageRegion	None	21.7.2020	22.7.2020
3.3	Adding Interaction Module and declare that in application Component	None	22.7.2020	23.7.2020
3.4	Add classifier isEquivalent	3.3	22.7.2020	24.7.2020
3.5	Map classifiers rules to DragDropModule using annotation class	3.4	24.7.2020	25.7.2020
3.6	Registering Interaction inside InteractionObjectTypeExtractorRepository	3.6	24.7.2020	25.7.2020
3.7	Low-fi : Adding Ratio Input Custom View	None	25.7.2020	28.7.2020
3.8	Low-fi : Adding Ratio Input View Part 2 : Add Ratio item View and its corresponding ViewModel	3.7	29.7.2020	5.8.2020
3.9	HighFi : Ratio Input Interaction	3.8	6.8.2020	12.8.2020
3.10	Additional Documentation,Wiki's for adding Interactions etc	None	13.8.2020	17.8.2020

Optional Sections

Future Work

I would like to work on improvising the architecture of the project. The current implementation of Clean Architecture has some flaws in it and also does not completely align with the rules/guidelines of the architecture pattern, so i would love to improve that part also i would love to integrate things like Kotlin Multiplatform(KMP) in order to support iOS devices as well other devices.

Documentation Changes

List the changes (if any) that should be made to the Oppia wiki to help support this feature or provide additional guidance to maintainers of Oppia instances. Types of projects that should include documentation changes:

- Complex migrations that require custom steps
- Infrastructural changes that require updates to the developer-facing instructions (e.g. new APIs added to our Protractor testing library that can help improve the reliability of e2e tests)

Additional Info

This doc is in the response of mail received on Tues April 7 which was regarding some clarifications from original proposal [Additional interaction types](#).

TECHNICAL DESIGN

- One of the proposed libraries is unlicensed. This may lead to significant changes in the implementation strategy. Please analyze this and provide a revision of the corresponding section.

In this original proposal i mentioned two libraries [PhotoViewAttacher](#) and [ClickableImageAreas](#) out of these two i'll be using only the first one which is licensed and maintain by author working at google. There was some misunderstanding because the second one was only my inspiration i will be creating that one from scratch and regarding the same i mentioned all the classes that would be required for creating the library.

- Please explain how you would ensure that the drag & drop functionality is accessible.

To make drag drop functionality more accessible specially with talkback enabled i've thought of to implement given below

The elements can contain a number

1. xxx
2. yyy
3. zzz

That way the screen reader will read out the position of the element and instead of drag and drop, will have to use voice commands instead "switch 1 and 3" will swap their positions like that or "select 3", "move up".

- The backend design for the Ratio interaction should use a ratio object instead of string parsing. Please provide an updated version of this section.

Based upon the discussion with Ben Hemming below is the new proposed Ratio Object and its protoBuf Model for android.

```
class Ratio(BaseObject):
    """Ratio class."""

    description = 'A ratio list.'
    default_value = [1,1]

    SCHEMA = {
        'type': 'list',
        'items': PositiveInt.SCHEMA
        'validators': [{
            'id': 'has_length_at_least',
            'min_value': 2
        }]
    }

message Ratio {
    repeated uint32 ratio_count = 1;
}
```

- Do we need to implement gcd ourselves for the isEquivalentTo case, or would it suffice to use the BigInteger gcd function?

The current BigInteger gcd function will not work for the ratio case as it takes up only two numbers and in our case function should take a list of numbers or should have variable length argument function.

- Please add more details on the testing approach. This should include an explanation of the various unit tests that would be required (especially for the classifiers in Android -- there aren't existing tests for these, but we should add some for the new interactions).

As in this project majority of the work will be around interaction so therefore most of the unit testing will be for newly introduced classifiers which will be part of domain module. In Domain module there are different classifiers for every interaction and every classifier has some set of rules to check the answer we will be writing test to check these rules we will use Robolectric tests. Some example of test are given below

- Ratio

```
val interaction = Interaction.newBuilder()
    .setId("Ratio")
```

```

.addAnswerGroups(AnswerGroup.newBuilder()
    .addRuleSpecs(RuleSpec.newBuilder().setRuleType("ListSizeEquals").putInput("r",
TEST_RATIO_0))
    .setOutcome(OUTCOME_0))
.addAnswerGroups(AnswerGroup.newBuilder()
    .addRuleSpecs(RuleSpec.newBuilder().setRuleType("isEquivalentTO
").putInput("r", TEST_RATIO_0))
    .setOutcome(OUTCOME_1))
.setDefaultOutcome(DEFAULT_OUTCOME)
.build()

```

- Drag, Drop and Sort

```

val interaction = Interaction.newBuilder()
    .setId("DragDrop")
    .addAnswerGroups(AnswerGroup.newBuilder()
        .addRuleSpecs(RuleSpec.newBuilder().setRuleType("isCorrectOrder").putInput("d",
TEST_DRAG_0))
        .setOutcome(OUTCOME_0))
    .setDefaultOutcome(DEFAULT_OUTCOME)
    .build()

```

- ImageSelection

```

val interaction = Interaction.newBuilder()
    .setId("ImageSelection")
    .addAnswerGroups(AnswerGroup.newBuilder()
        .addRuleSpecs(RuleSpec.newBuilder().setRuleType("isValidRegion").putInput("i",
TEST_IMAGE_0))
        .setOutcome(OUTCOME_0))
    .setDefaultOutcome(DEFAULT_OUTCOME)
    .build()

```

MILESTONES

- If selected, you will probably need ~1 week at the beginning of the Community Bonding Period to finalize any outstanding questions in the proposal. We suggest not scheduling any implementation tasks for that week.

Due to covid-19 there is no confirmation about my college exams once there is a tentative for my exams i will be changing my milestone accordingly but till then i would like to start as soon as possible so that i don't fail to match my deadlines if my exams happen during the period of gsoc.

OTHER NOTES

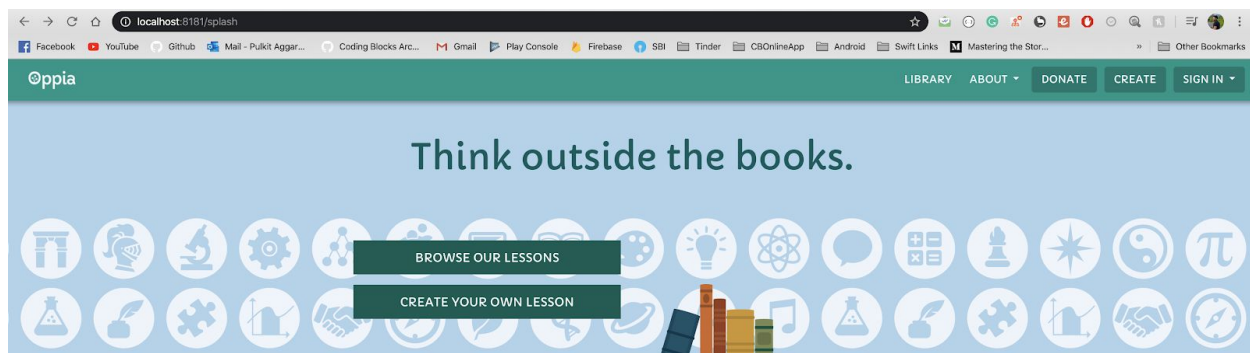
- The author should verify that they can run the backend tests & the Oppia dev server locally.

I was able run successfully run all the test on my local machine

```
SUCCESS scripts.create_topological_sort_of_all_services_test.TopologicalSortTests: 3 tests (2.1 secs)
SUCCESS scripts.run_e2e_tests_test.RunE2ETestsTests: 39 tests (21.8 secs)
SUCCESS scripts.release_scripts.cut_release_or_hotfix_branch_test.CutReleaseOrHotfixBranchTests: 20 tests (10.1 secs)
SUCCESS scripts.release_scripts.update_configs_test.UpdateConfigsTests: 12 tests (5.0 secs)
SUCCESS scripts.release_scripts.restore_backup_test.RestoreBackupTests: 7 tests (4.1 secs)
SUCCESS scripts.release_scripts.gcloud_adapter_test.GcloudAdapterTests: 11 tests (5.4 secs)
SUCCESS scripts.release_scripts.wrap_up_release_test.WrapReleaseTests: 9 tests (4.1 secs)
SUCCESS scripts.release_scripts.initial_release_prep_test.InitialReleasePrepTests: 7 tests (4.1 secs)
SUCCESS scripts.release_scripts.update_indexes_test.UpdateIndexesTests: 3 tests (2.8 secs)
SUCCESS scripts.release_scripts.deploy_test.DeployTests: 38 tests (17.5 secs)
SUCCESS scripts.release_scripts.update_changelog_and_credits_test.ChangelogAndCreditsUpdateTests: 23 tests (9.5 secs)
SUCCESS scripts.release_scripts.generate_release_updates_test.GenerateReleaseUpdatesTests: 9 tests (4.7 secs)
SUCCESS scripts.release_scripts.generate_release_info_test.GenerateReleaseInfoTests: 18 tests (7.5 secs)
SUCCESS python_utils_test.PythonUtilsForPython2Tests: 2 tests (2.4 secs)
SUCCESS python_utils_test.PythonUtilsForPython3Tests: 2 tests (0.0 secs)
SUCCESS python_utils_test.PythonUtilsTests: 20 tests (9.8 secs)

Ran 5114 tests in 811 test classes.
All tests passed.
```

I was successfully able to run Oppia dev server locally



- We would like to see one PR from the author to oppia/oppia that includes Python changes.

Open Pull Request - <https://github.com/oppia/oppia/pull/9030>

Issues that i'll work on - <https://github.com/oppia/oppia/issues/9063>

