Problem: Create a simple end-to-end web application as per the instructions in this document.

Steps:

1. Choose a web framework of your preference (e.g., Python-flask) and front-end framework (e.g., AngularJS + Bootstrap).
2. Implement REST API functionality as described in **BACK_END_DEVELOPMENT** section.
3. Finally, build/implement web-interface described in **FRONT_END_DEVELOPMENT** section**.**

All the required data and web-interface design PSDs are in the same folder.

Expected output:

1. Clean code with comments
2. a small writeup on how to deploy and execute the application.

What we are looking for?

Ability to design, write clean and well documented code.

BACK_END_DEVELOPMENT

Implement the below API with web-server of your choice. The objective of the API is to accept a set of numeric parameters as a request, retrieve and generate collections of data (each collection is an individual group/cluster) from the data file (*images_list.csv*) as per the input parameters and respond to the request in JSON format as per the API.

Notes:

- *images_list.csv* has two columns image_url and image_id.
- Implement a function with name *cluster_images* that
    a. takes *cluster_count* from api_request as input
    b. randomly split images data into *cluster_count* number of groups
    c. prepare the json response using the groups in step (b) and the response json structure described below **API Response**
- The *centre_image* of clusters can be set as any one of the *image_id* in the group

API

Request

Get image clusters from images present in database/list on web-server

Method	URL
POST	web-server-url/cluster_images/

Type	Params	Values
POST_PARAM	cluster_count	int
POST_PARAM	parameter_shape (0-1)	fraction
POST_PARAM	parameter_color (0-1)	fraction
POST_PARAM	parameter_pattern (0-1)	fraction
POST_PARAM	parameter_objects (0-1)	fraction
POST_PARAM	parameter_activities (0-1)	fraction

Response

Some Notes: **image_id_x** is integer id present in *images_list.csv*
centre_image for this test can any one of the *image_id* present in group/cluster
cluster_id is random number unique for each group/cluster of data points
cluster_name should be empty string ""
data_size is total number of images in the dataset (here *images_list.csv)*
clustered_count is total number of images in the individual group/cluster
coverage is percent of images present in a particular group/cluster out of total number of images

Status	Response

200	{
	"all_images" : {
	"image_id_1" : {
	"url" : "https://scontent-lht6-1.cdninstagram.com/vp/73a1c0b01ded4bfcd4bf7fb3aeebb848/5CBAA2CB/t51.2885-15/e35/49308640_293102291564594_8115251538130295912_n.jpg?_nc_ht=scontent-lht6-1.cdninstagram.com",
	} ,
	"image_id_2" : {...},
	…..,
	"image_id_n" : {...}
	},
	"cluster_group_summary" : {
	"data_size" : 1000,
	},
	"clusters_list" : List[

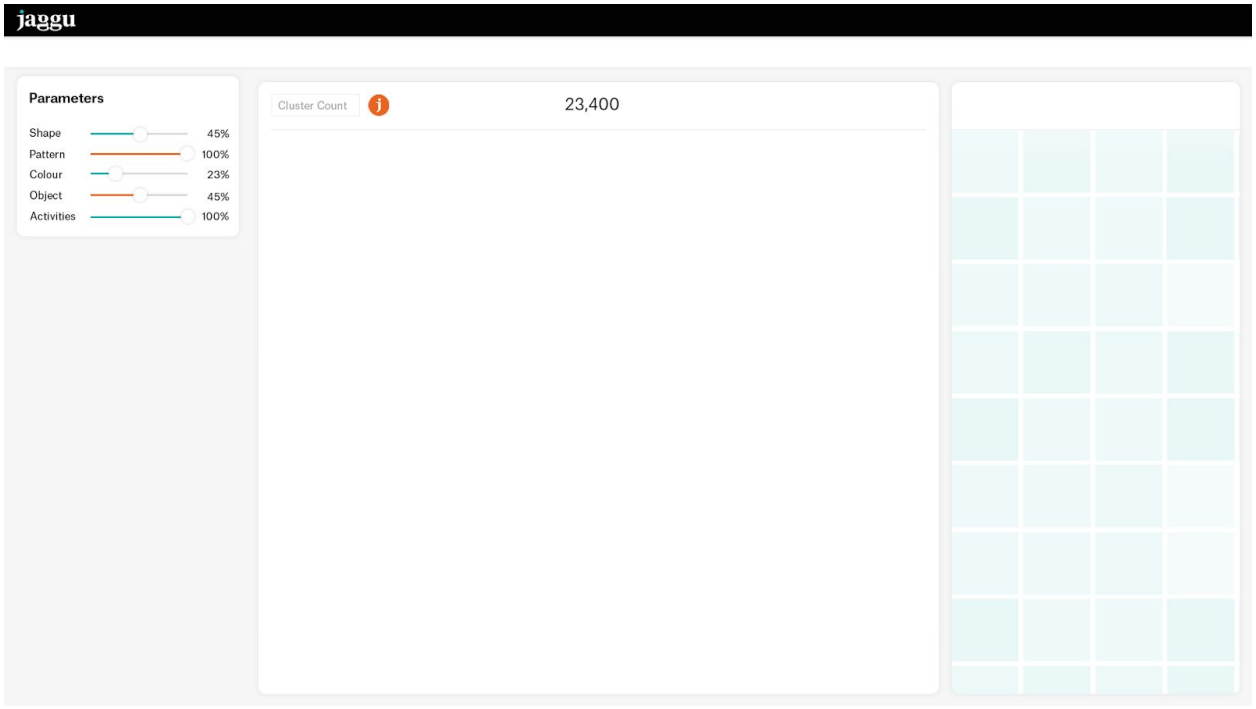
	{
	"centre_image" : image_id_3,
	"cluster_id" : 8,
	"cluster_images" : List[image_id_1, image_id_2, image_id_3, …],
	"cluster_name" : "",
	"cluster_summary" : {
	"clustered_count" : 35,
	"coverage" : 7.45,
	},
	},
	….
	]
	}

404	{"error":"Parameters are missing."}
500	{"error":"Something went wrong. Please try again later."}

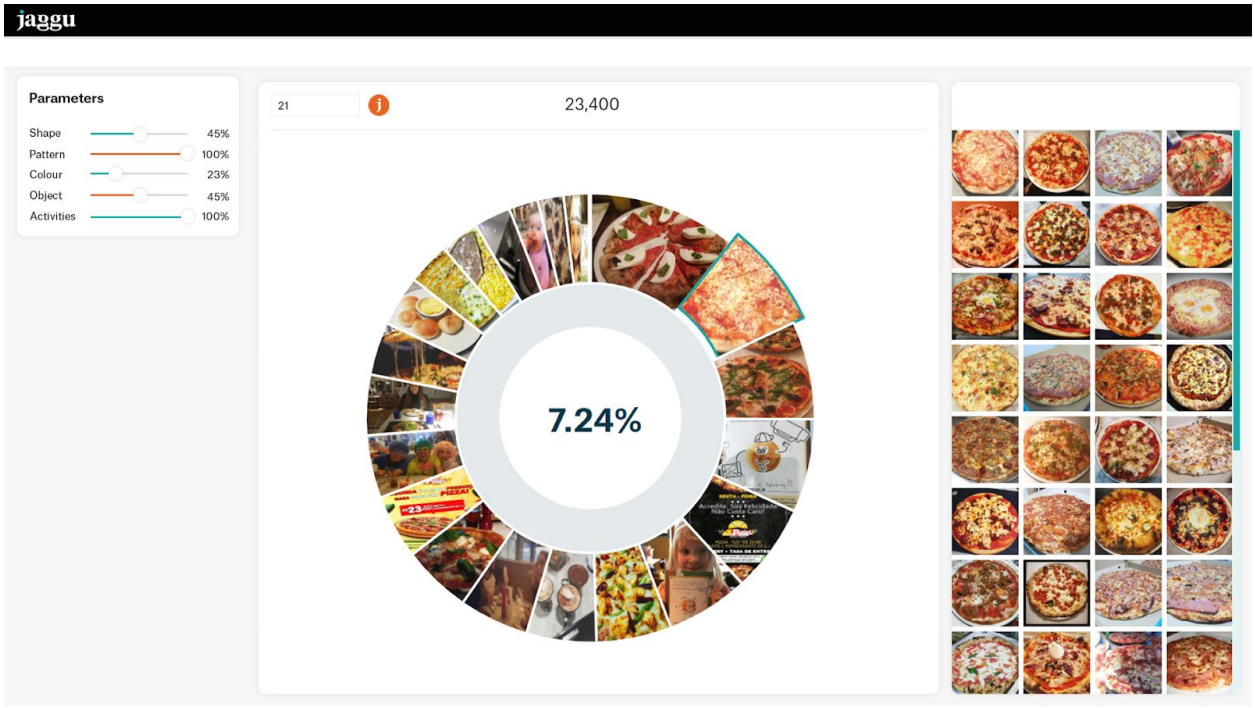
FRONT_END_DEVELOPMENT

Create a responsive web interface to provide a clustering interface. The corresponding PSDs are found in the test folder.

Landing Page: The input `cluster count` and the `Parameters`.



The view after api-call-return constructed using the API response should look like below:



Some Notes:

- The wheel contains all the clusters/groups with *centre_image* (from api-call) being the image displayed on the arc.
- The centre of wheel displays *coverage* for selected group/cluster
- The pane on right-most displays all images for the selected group/cluster.
- The image urls are public web urls -- will be displayed using the image URLs in the API response.