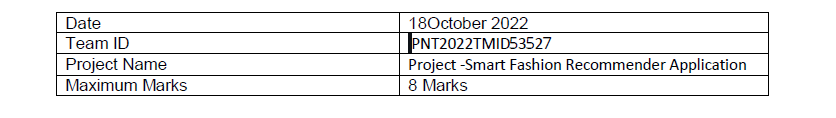
PROJECT PLANNING TOOL:



PYTORCH:

We will make use of transfer learning, approximate nearest neighbors, and embeddings centroid detection in PyTorch to build our recommender.

Once we apply a transformation to convert the objects into embeddings, we now limit the number of elements in the array representing each item (the limit is 2 in this example) on a continuous scale, and the values have a relationship-based meaning. Objects close to each other based on similarity (dot product) are highly related.

SHORT SAMPLE CODE:

def train\_model(data, pretrained\_model, model\_metrics):

learner = cnn\_learner(data, pretrained\_model, metrics=model\_metrics)

learner.model = torch.nn.DataParallel(learner.model)

learner.lr\_find()

learner.recorder.plot(suggestion=True)

return learner

pretrained\_model = models.resnet18

model\_metrics = [accuracy, partial(top\_k\_accuracy, k=1), partial(top\_k\_accuracy, k=5)]

learner = train\_model(data, pretrained\_model, model\_metrics)

def load\_learner(data, pretrained\_model, model\_metrics, model\_path):

learner = cnn\_learner(data, pretrained\_model, metrics=model\_metrics)

learner.model = torch.nn.DataParallel(learner.model)

learner = learner.load(model\_path)

return learner

pretrained\_model = models.resnet18

model\_metrics = [accuracy, partial(top\_k\_accuracy, k=1), partial(top\_k\_accuracy, k=5)]

model\_path = "/content/gdrive/My Drive/resnet18-fashion"

learner = load\_learner(data, pretrained\_model, model\_metrics, model\_path)

# takes time to populate the embeddings for each image

# Get 2nd last layer of the model that stores the embedding for the image representations

# the last linear layer is the output layer.

saved\_features = SaveFeatures(learner.model.module[1][4])

\_= learner.get\_preds(data.train\_ds)

\_= learner.get\_preds(DatasetType.Valid)