Executive Summary or Abstract:

TrackAtool is a wearable-device which uses a cloud-based [deep-learning](https://en.wikipedia.org/wiki/Deep_learning) framework to help human-memory to recall the location of their day-to-day objects. This tracks all objects around, providing a simple-yet-efficient mechanism to solve problem that is so prevalent in dementia people. This mechanism, unlike a tag-based object identification, is not limited to objects that have been specifically tagged. It is placed over the clothing, Captures images as user moves around and processes them to extract the objects within them.

Introduction:

* Once I had misplaced my keys which I searched a lot.
* By recalling,
  + when I have last used the keys,
  + I was able to track the keys in around 10 -15 minutes.
  + from where I have taken and
  + Where I have gone carrying those keys.
* I have discussed this thing to my friend who is pursuing MBBS, in a casual chat.
* What she said was, “you are lucky that you are having the recalling power where as she met a person who she met in her college is not able to recall the incidents that happened recently.”
* The reason is that the person is suffering from **Dementia**.
* So, what if dementia people misplaced their daily belongings?
* We searched about this and found that there are no reliable devices which will keep a track of the daily belongings one uses.

Objective:

We aim to make dementia people independent from others, while searching their belongings in day-to-day life.

Need Statement:

A way to solve forgetfulness of daily belongings by dementia people to ease their daily life.

Mission Statement:

Customer Needs:

From our user study, we have derived the following customer needs

Primary: (must)

1. Objects that would be covered
   * Medicines, keys, spectacles, books, cards, wallet, etc.
2. Work places that need be covered
   * Home, office (for job holders), School (for students).
3. Range that would be covered is 2-3 m.
4. Can be used Independently

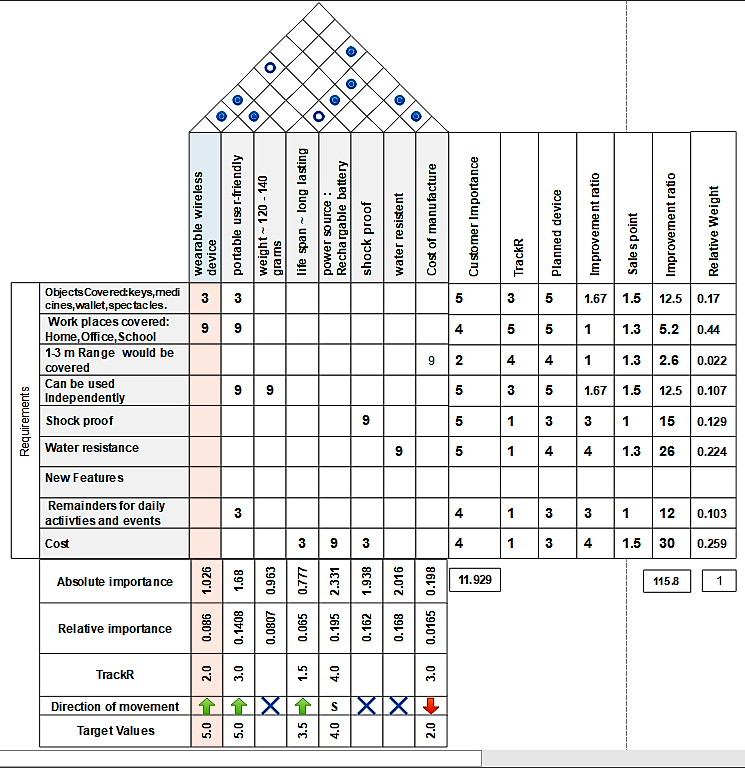
Secondary: (may/may not)

* A reminder for daily activities and medications.

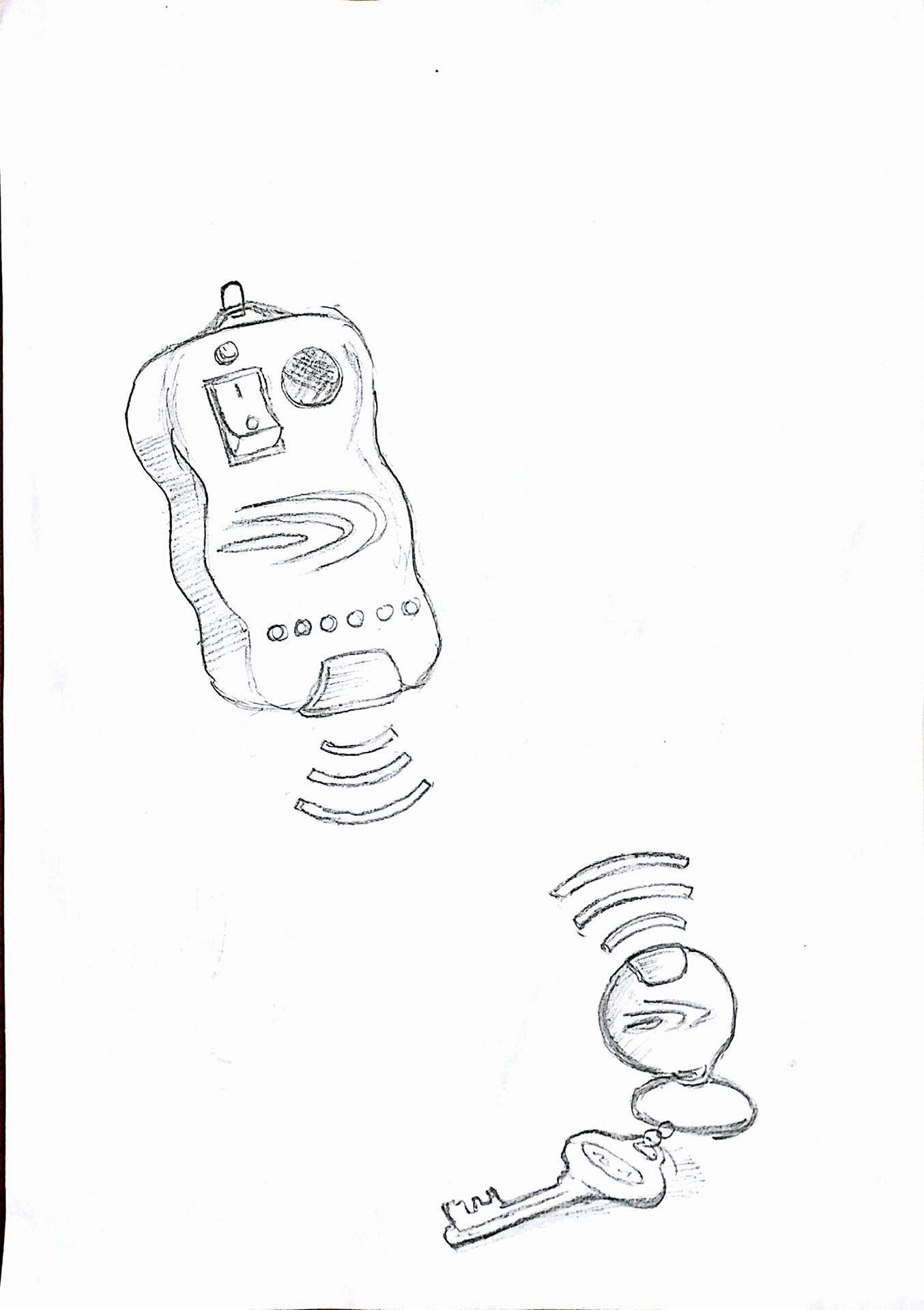
Product Specifications:

* Wearable Wireless device
* Portable and user-friendly
  + Weights around 120-140 grams
* Life span:- long lasting
* Power source – rechargeable battery
* Shock proof
* Water resistant
* Cost approximately ~ ₹3000 - ₹5000

QFD:

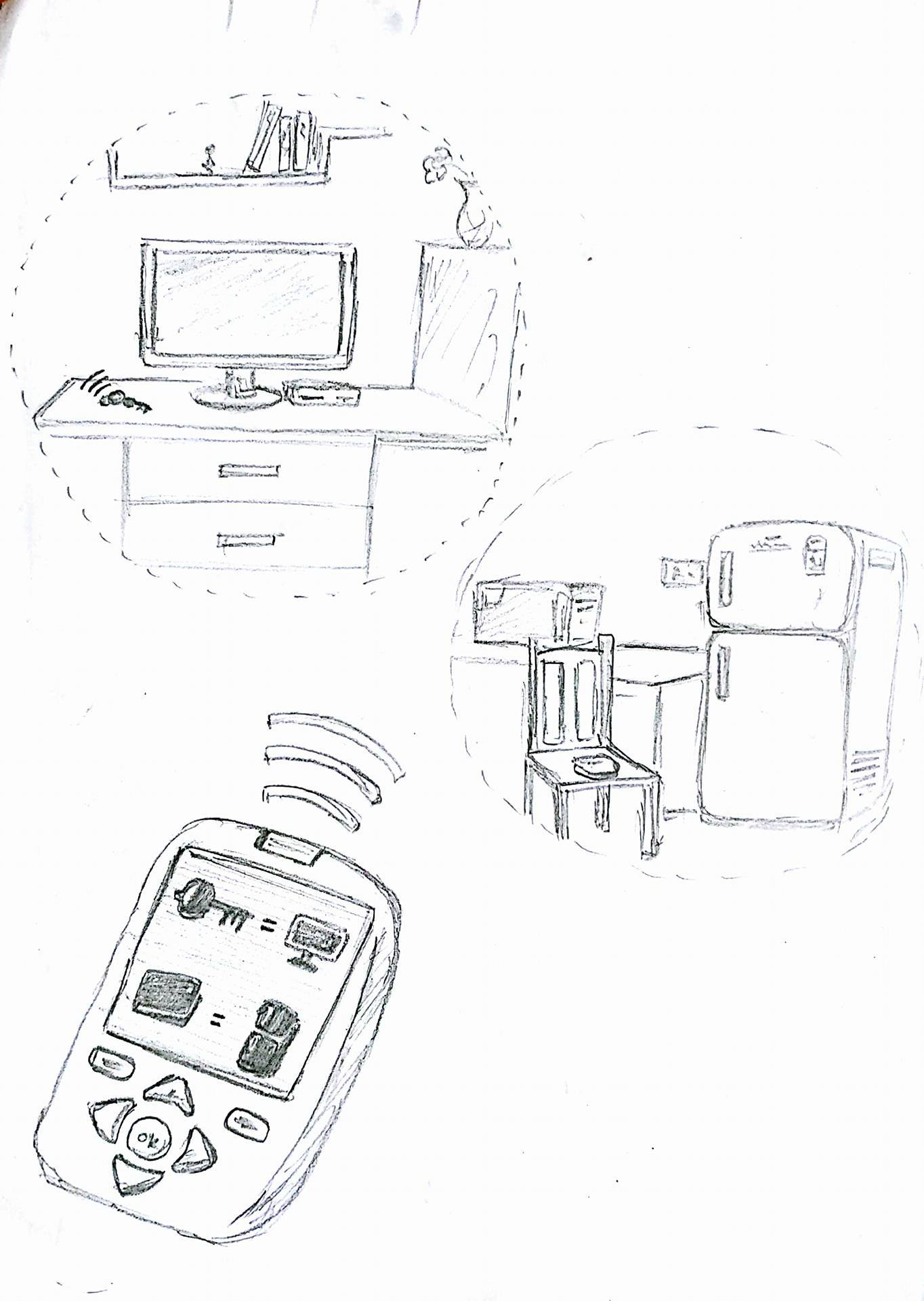


Concept Generation:



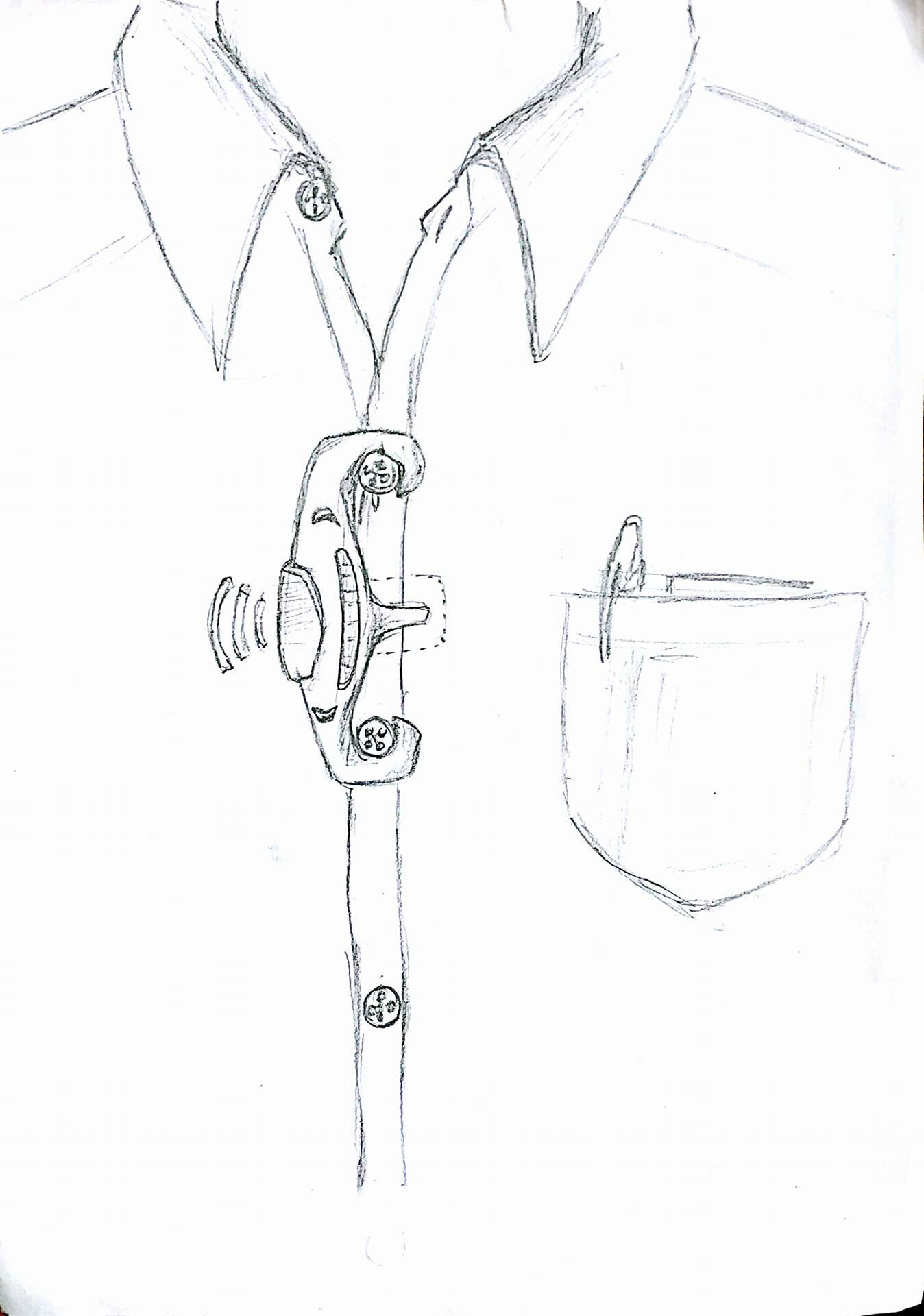
Concept 1:

* Every object is attached with a chip.
* This chip emits a particular frequency.
* Device detects these particular frequency & helps to locate things.



Concept 2:

* Like a broadcasting telephone a range can be created for a particular reference object.
* Every object which comes inside this range is identified automatically.



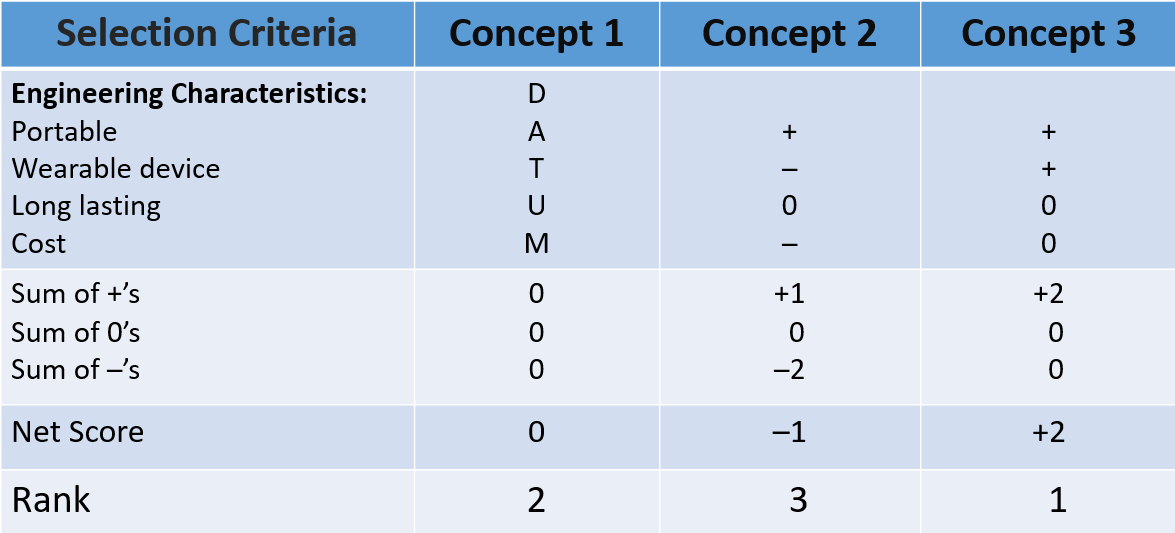
Concept 3:

* A wearable device keeps tracks of all the objects around us
* It stores the location of the corresponding object
* When asked for the object, it shows the location w.r.t reference object

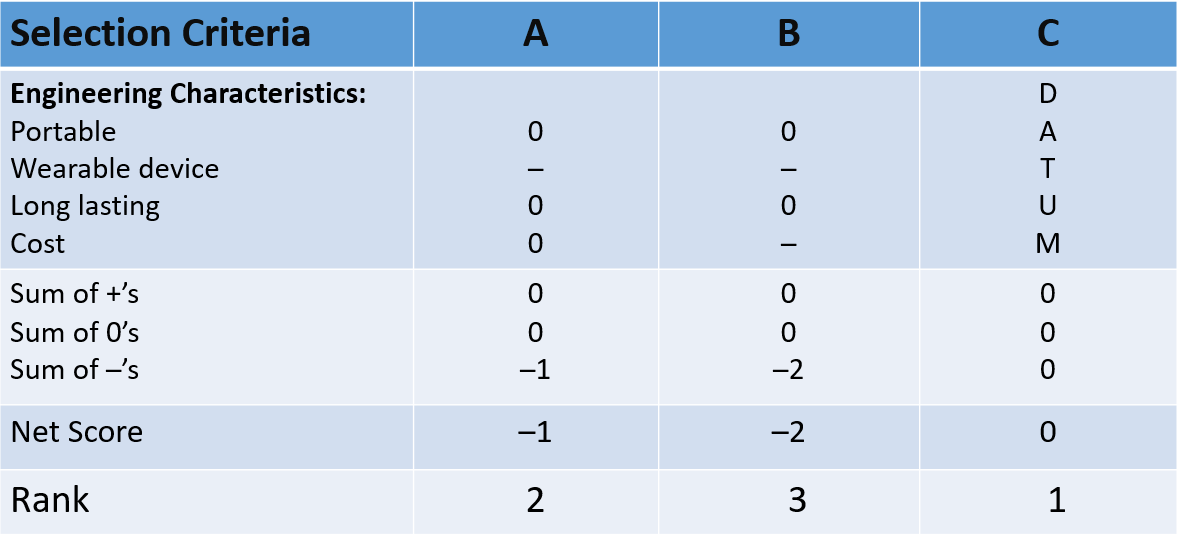
Concept Evaluation using Pugh’s method:

Relative Ranking:

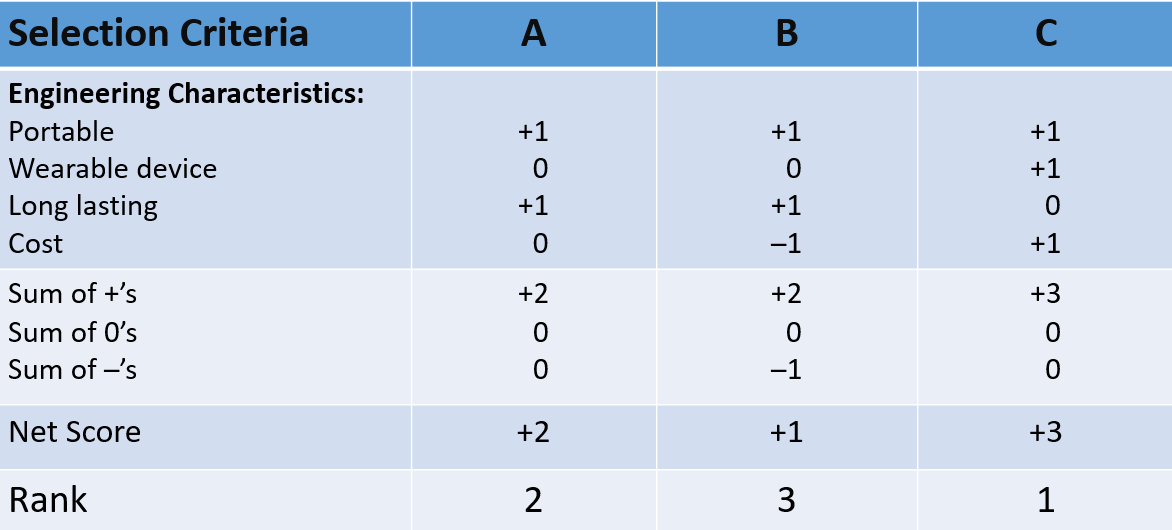
* Treating concept 1 as DATUM (reference)



* As concept 3 ranked 1 now we treat concept 3 as DATUM



Absolute Ranking:



* From Pugh’s method, Concept 3 is ranked first.

Algorithms/Computational Implementation:

The algorithm contains 3 parts:

1. For Storing Input Images.
2. Reading and Processing Image.
3. Identification of Images w.r.to stored images.

Pseudo code for part-1:

1. function [tr\_set,test\_set] = prepareInputFiles(dsObj)
2. image\_location = fileparts(dsObj.Files{1});
3. imset = imageSet(strcat(image\_location,'\..'),'recursive');
4. [tr\_set,test\_set] = imset.partition(700);
5. test\_set = test\_set.partition(200);
6. end

Pseudo code for part-2:

1. function Iout = readAndPreprocessImage(filename)
2. I = imread(filename);
3. if ismatrix(I)
   * I = cat(3,I,I,I);
4. end
5. end

Pseudo code for part-3:

1. title(sprintf('Best Guess: %s; Actual: %s',char(label),testSet.Labels(randNum)))
2. testFeatures = activations(convnet, testSet, featureLayer, 'MiniBatchSize',32);
3. predictedLabels = predict(classifier, testFeatures);
4. testLabels = testSet.Labels;
5. confMat = confusionmat(testLabels, predictedLabels);
6. confMat = bsxfun(@rdivide,confMat,sum(confMat,2));
7. mean(diag(confMat))
8. maxCount = size(testSet.Files,1);
9. randNum = randi(maxCount);
10. newImage = testSet.Files{randNum};
11. img = readAndPreprocessImage(newImage);
12. imageFeatures = activations(convnet, img, featureLayer);
13. label = predict(classifier, imageFeatures);
14. testSet.Labels(randNum)
15. imshow(newImage);
16. if strcmp(char(label),char(testSet.Labels(randNum)))
    * 1. titleColor = [0 0.8 0];
17. else
    * 1. titleColor = 'r';
18. end