Monument Recognition

By,

Team -5

Panja,Kumara Satya Gopal(kpqmd@mail.umkc.edu)

Linga,Sivarama Krishna(slhx4@mail.umkc.edu

Vikesh Padarthi (vpn7d@mail.umkc.edu)

Vundela,Karthik Reddy(kvxc5@mail.umkc.edu)

Project Report -2

Big Data Analytics and Applications

1 Introduction:

Nowadays finding the matching of images in order to establish a measure of similarity is a major problem. Here we introduce a "Monument Recognition System". It takes monument image as input from user and gives description about that images as output. For this first we want create our own data set for different monuments. Based on different algorithms, we can find input belongs to which category. Based on that monument category and apis we can give description about that monument.

2 Project Objectives:

2.1 Significance:

Whenever we visit a place we not only want to take the pictures in front of it but we would like to know history about that place. So instead of relying on some guide or go through the Google and search, a simple website which will help to know about the place in details in both text and audio format. Our objective is to develop a simple website where for the website we will accept input as a image which might be a statue or a building. we will learn about the image and know what it is like statue of liberty and we will get the details of that image like who developed and where it is etc and we will display about it. And we want to turn this text into voice and user should be able to listen to it.

2.2 Features:

- User can open the application on web browser and see the home page. On clicking on home page user will see signin and signup pages.
- User needs to create an account by clicking on signup page.
- User can login by using his credentials
- On Successful login we navigate user to main page.
- In main page, user can upload an image.
- User can learn about the monument
- User can find related places
- User can hear the speech about monument.

3 Approach

3.1 Data Sources:

Here we can create our own data set for different monuments. And we can use google and some apis for retrieving data about monuments**.**

3.2 Analytical Tools:

Clarifai API:

This is used to detect what is present in a given. We use the monument dataset and the Clarifai API for detecting what is present in the image.

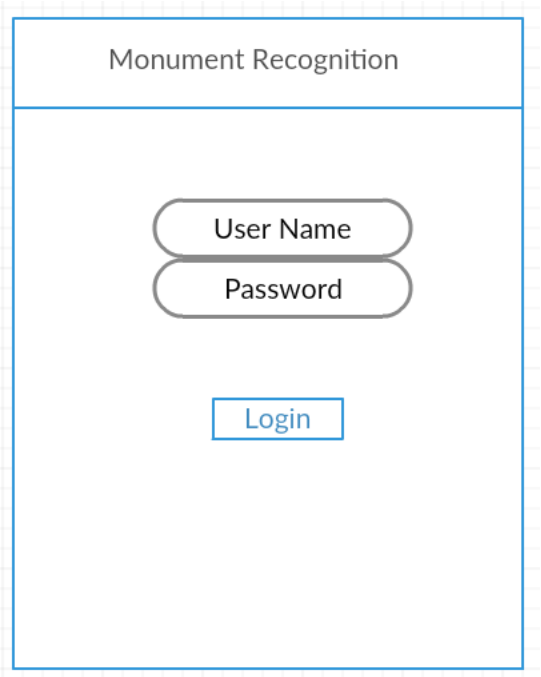
Spark(ML Library):
The Spark machine learning library is used for training the the image data set with various machine learning algorithms such as k-means etc which are in-built in the Spark ML library and this can later be used to test the images after we get a trained model.
Tensor Flow:
Tensor Flow is also being used for the image classification purpose.

3.3 Analytical Tasks

Initially the images are used for building the model. This is called the training phase. After building the model testing can be done to get the information about image. The input image is first analysed for extracting the key features. Then the histograms are built from these key features which represent the bag of words present in the image. This is then used as input for training the model. After the model is finally is tested with various sets of inputs and the process is repeated by tuning the parameters until desired accuracy and acceptable error rate, precision,recall and f-measure are obtained. After the model gets ready it can be used for the end-user app. In addition this the Clarifai API and tensor flow are also used to get better results.

3.4 Expected Inputs/Outputs:

When we finish developing this system, it should be able to provide movies recommendation to users. Based on user preferences, out system will recommend movies to users.

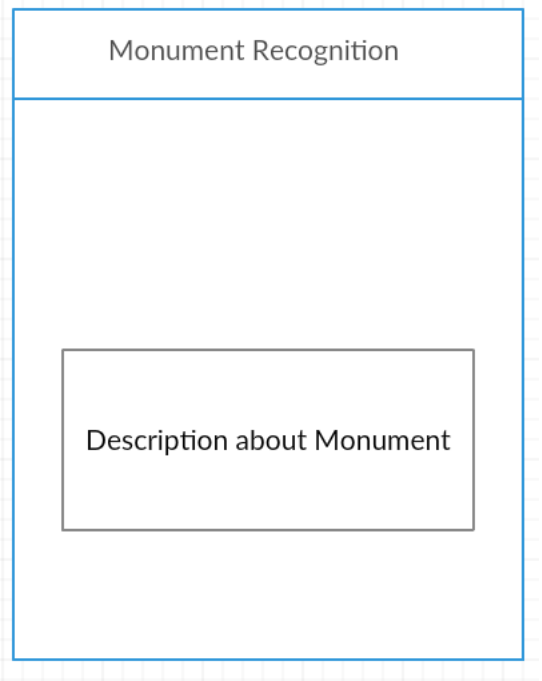


When the user opens our application, first it shows login page. User need to create an account. By using the user credentials, he can login into the application. On successful login, it will navigate into main page.

Fig 1

User can upload monument image

Fig 2



User can see the description about that monument

Fig 3

3.5 Algorithms

There are several algorithms that can be used for the purpose of analysing the data sets.
However, based the application appropriate algorithms need to be chosen. For example, Naive Bayes performs better when the data set is a text data. On the other hand image data gets better results when k-means is used. Hence we used K-means for classifying the image.
Initially the number of clusters and iterations are predefined and the given dataset is clustered into different clusters based on the common attributes. Then the model is built based on the training data. After that testing can also be done.
We also used random forest algorithm for image classification. Random forest proves to be more advantageous when compared to decision tree as decision tree involves only one tree whereas random forest is an ensemble of decision trees in parallel thereby giving better accuracy and lesser error rate.

4 Related Work:

4.1 Open Source Projects:

The image classification model in the project has been inspired from the model below. It has several programs for classification of images.

https://github.com/nteetor/Image-Classifier

A list of projects along with guidelines to do them has been provide in the below link which were all used as a part of final year projects
http://www.di.ens.fr/willow/teaching/recvis09/final_project/

4.2 Literature Reviews:

Two pairs also helped in getting an idea of what image classification is about. The first gives a clean account of the process that has to be adopted for doing any image classification.

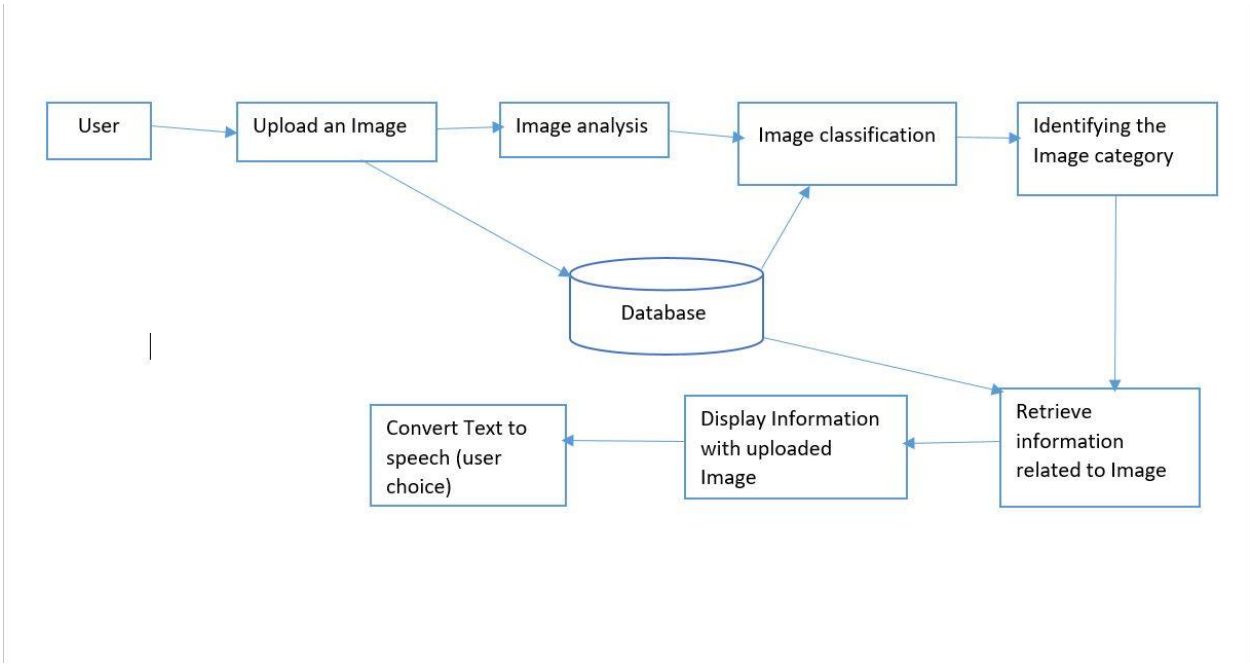
The second one is more subjective I.e. it provides a way of distinguishing images (which is explicit from the name of the paper!). http://cs229.stanford.edu/proj2013/KrzesinskiWilder-ImageObjectClassification.pdf

https://sites.ualberta.ca/~bang3/files/DogCat_report.pdf

5 Application Specification:

5.1 Software Architecture:

5.2 Work flow Diagram:



5.3 Sequence Diagaram:

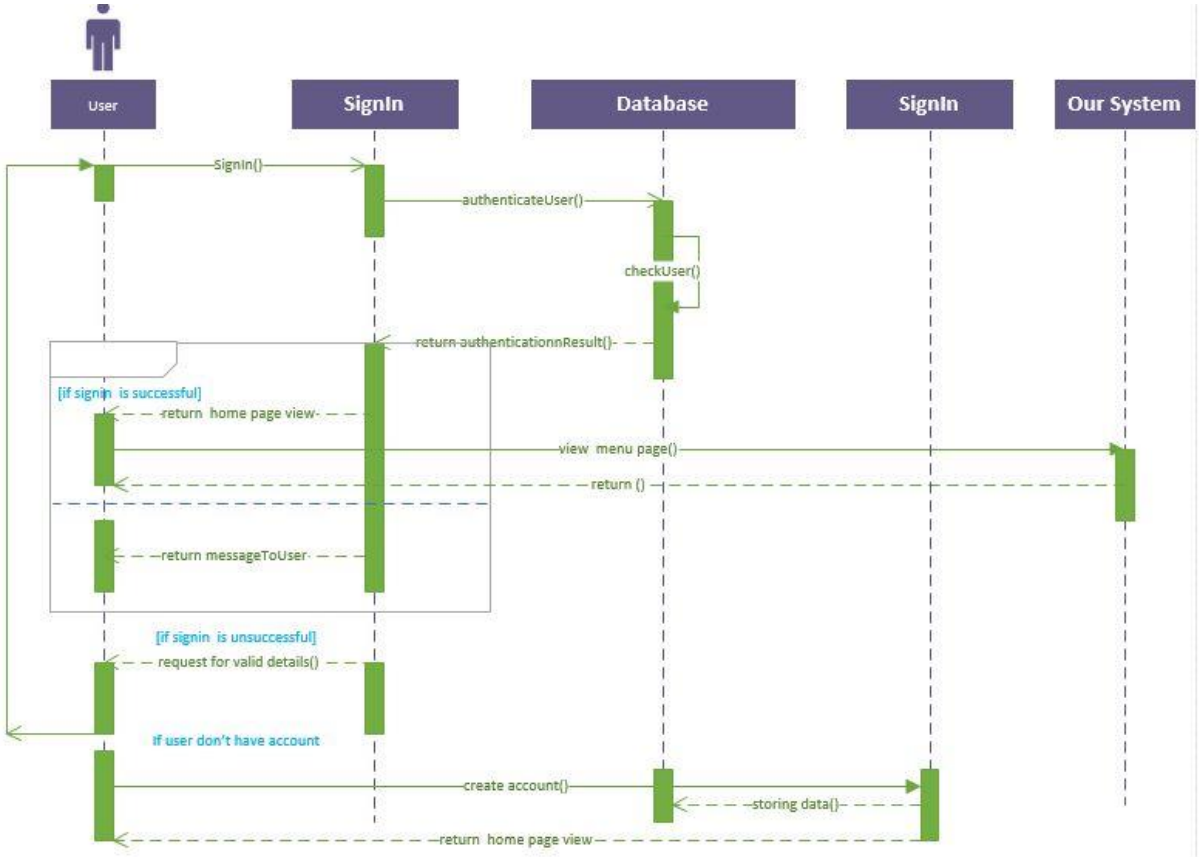


Fig 4

6. Implementation:

6.1 Clarifai API outputs:

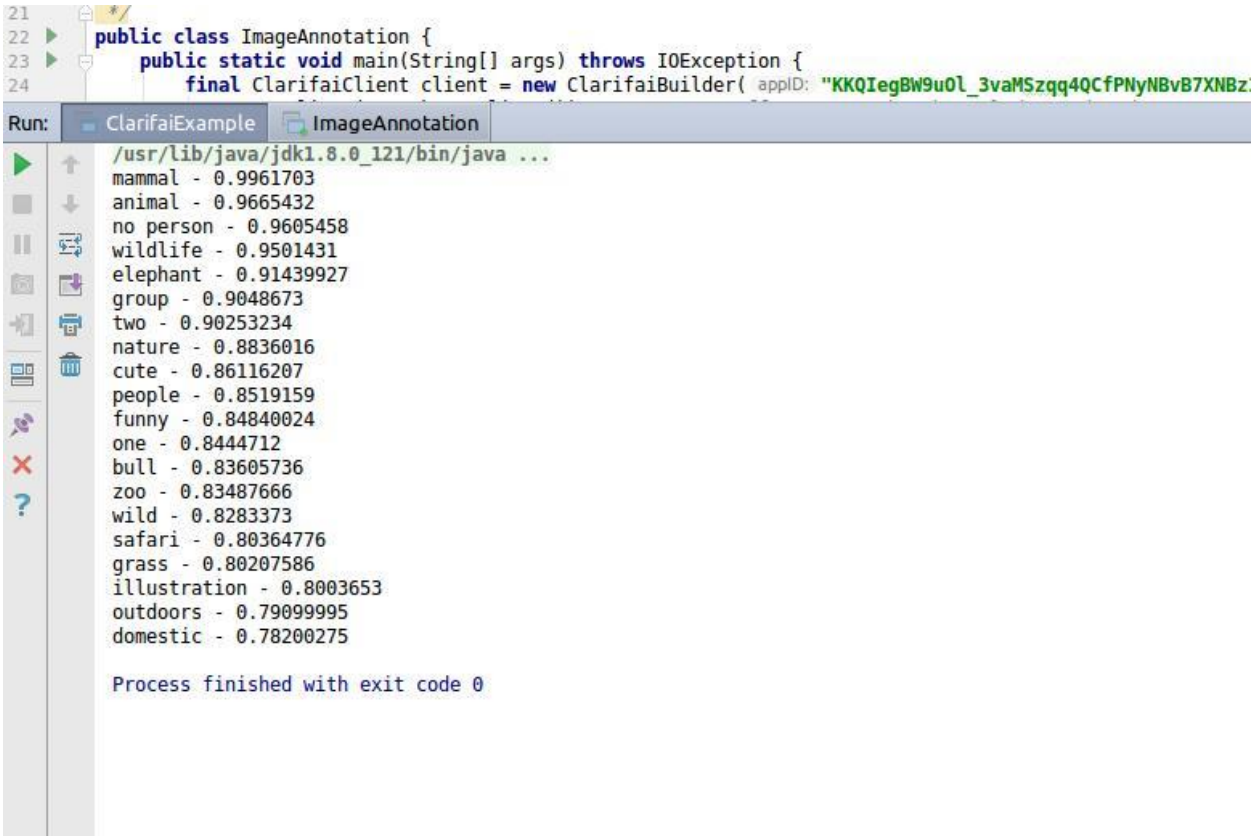


Fig 5

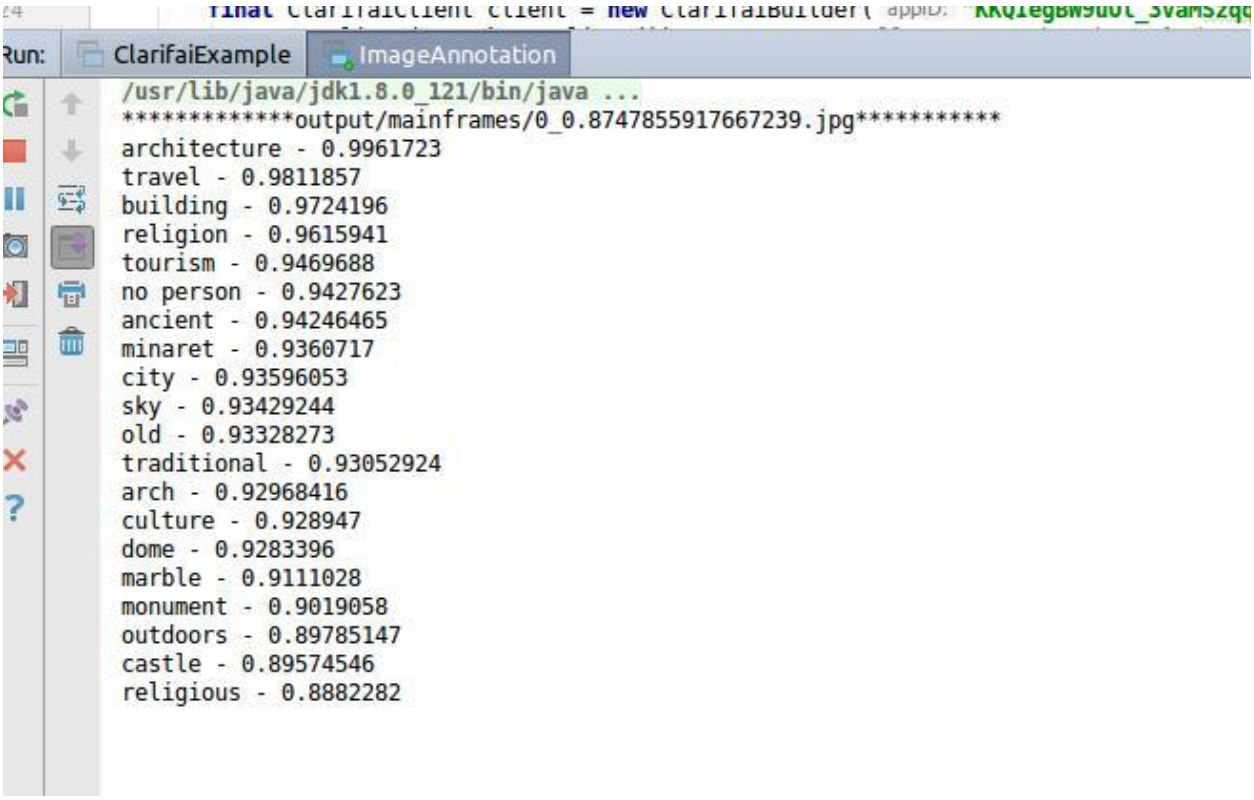

Run:
ClarifaiExample
ImageAnnotation
/usr/lib/java/jdk1.8.0_121/bin/java ...
**********output/mainframes/0_0.8747855917667239.jpg**********
architecture - 0.9961723
travel - 0.9811857
building - 0.9724196
religion - 0.9615941
tourism - 0.9469688
no person - 0.9427623
ancient - 0.94246465
minaret - 0.9360717
city - 0.93596053
sky - 0.93429244
old - 0.93328273
traditional - 0.93052924
arch - 0.92968416
culture - 0.928947
dome - 0.9283396
marble - 0.9111028
monument - 0.9019058
outdoors - 0.89785147
castle - 0.89574546
religious - 0.8882282

Fig 6


image0
dome
marble
tourism
travel
ancient
culture
city
castle
sky
building
religion
arch
traditional

Fig 7

6.2 Google Conversation API outputs:

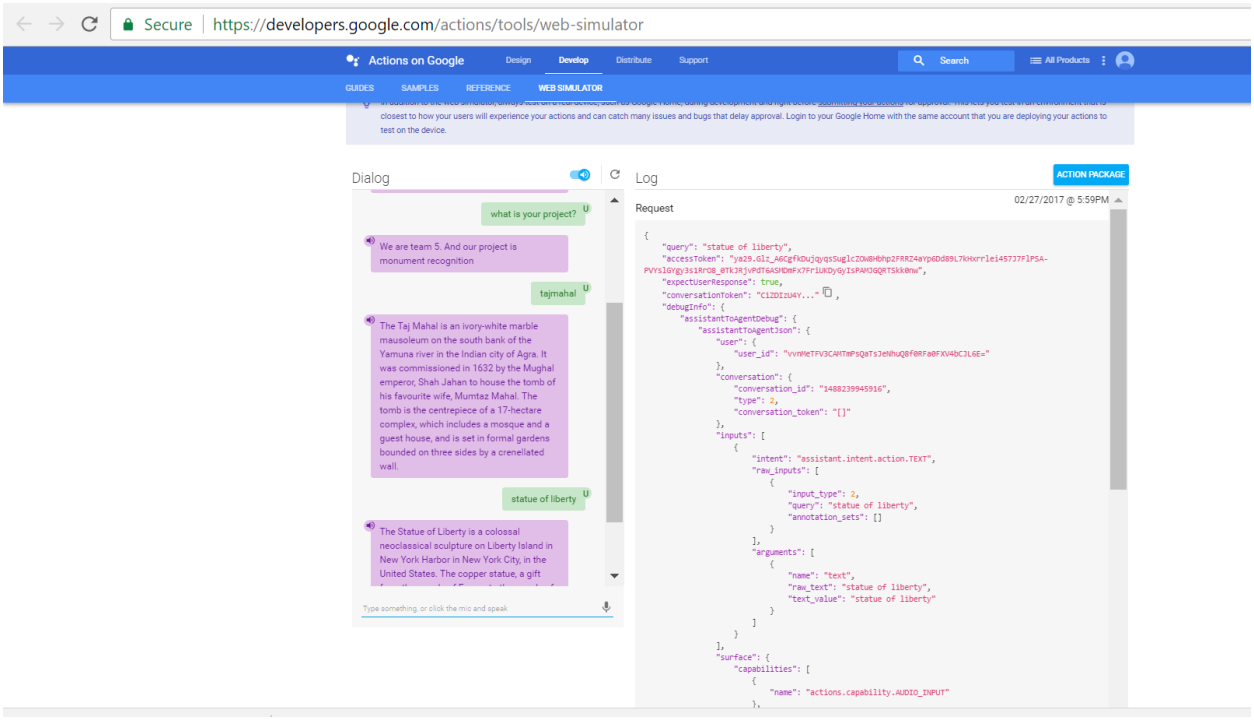


Fig 8

6.3 Application using Spark Program:

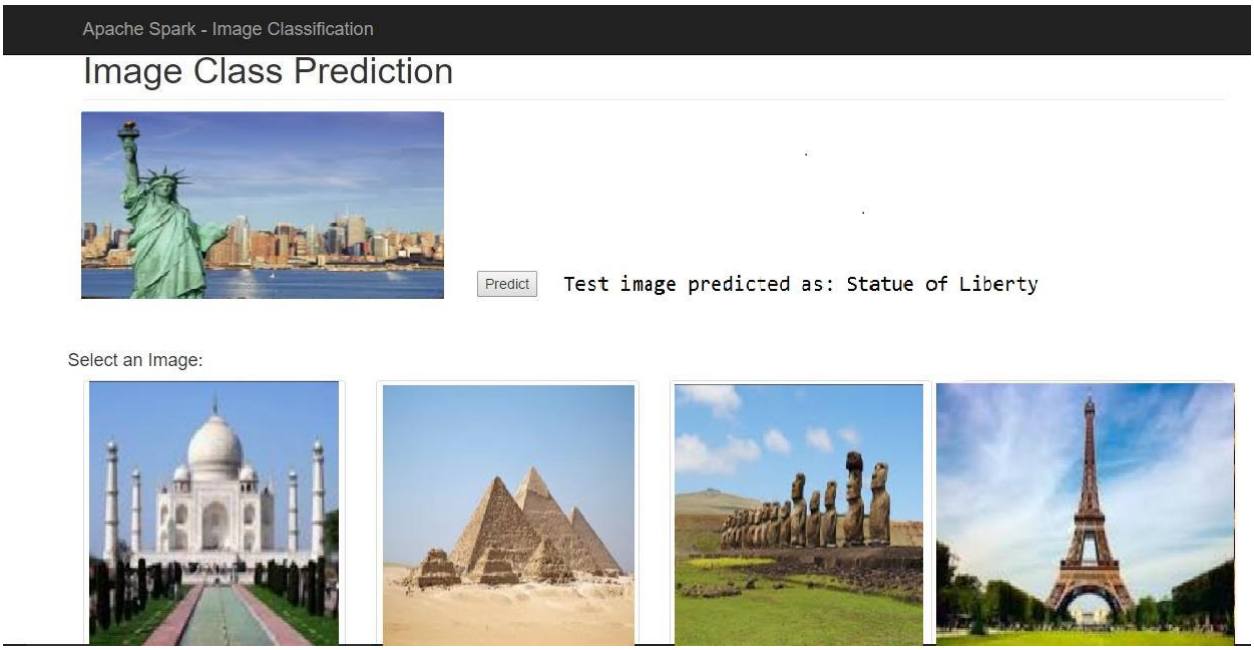


Fig 9

```
|==================== Confusion matrix ============================
10.0   1.0   2.0   1.0   7.0   2.0   2.0   3.0   4.0
6.0    1.0   7.0   2.0   8.0   0.0   1.0   7.0   1.0
3.0    6.0   8.0   1.0   10.0  0.0   1.0   4.0   0.0
2.0    0.0   3.0   5.0   9.0   2.0   4.0   3.0   5.0
5.0    3.0   5.0   4.0   8.0   1.0   2.0   1.0   4.0
4.0    2.0   3.0   4.0   5.0   3.0   7.0   2.0   3.0
2.0    4.0   0.0   2.0   8.0   4.0   6.0   1.0   6.0
3.0    1.0   5.0   7.0   6.0   2.0   2.0   2.0   5.0
1.0    1.0   2.0   3.0   5.0   2.0   11.0  1.0   7.0
0.16891891891891891
```

Fig 10

6.4 Tensor flow outputs:

Testing Results: Accuracy of 90%

```
W:0
b:0
W:0
b:0
[array([[ 0.,  0.,  0., ...,  0.,  0.,  0.],
       [ 0.,  0.,  0., ...,  0.,  0.,  0.],
       [ 0.,  0.,  0., ...,  0.,  0.,  0.],
       ...,
       [ 0.,  0.,  0., ...,  0.,  0.,  0.],
       [ 0.,  0.,  0., ...,  0.,  0.,  0.],
       [ 0.,  0.,  0., ...,  0.,  0.,  0.]], dtype=float32), array([-0.1227937 ,  0.34265223, -0.06169668, -0.11410887,  0.13021739,
        0.36001251,  0.04051079,  0.2901822 , -0.72707415, -0.14662132], dtype=float32), array([[ 0.,  0.,  0., ...,  0.,  0.,  0.],
       [ 0.,  0.,  0., ...,  0.,  0.,  0.],
       [ 0.,  0.,  0., ...,  0.,  0.,  0.],
       ...,
       [ 0.,  0.,  0., ...,  0.,  0.,  0.],
       [ 0.,  0.,  0., ...,  0.,  0.,  0.],
       [ 0.,  0.,  0., ...,  0.,  0.,  0.]], dtype=float32), array([-0.1227937 ,  0.34265223, -0.06169668, -0.11410887,  0.13021739,
        0.36001251,  0.04051079,  0.2901822 , -0.72707415, -0.14662132], dtype=float32)]
0.9004

Process finished with exit code 0
```

Fig 11

Cross Entropy Visualization

Fig 12

7 Project Management:

7.1 Contribution of Team Mates:

Class Id	Name	Responsilbilty
32	Panja, Kumara Satya Goal	Google Conversation API, Desgining UML diagrams and Expected outcomes, Application using Spark
21	Linga, Siva Rama Krishna	Clarifai API, Workflow Diagaram, Data collection, Tensor Flow
30	Padarthi, Vikesh	Architecture Diagram, User Interface, Testing, Dataset collection, Project Managment
43	Vundela, Karhik	Project Management, Gathering Requirements, Documentation, Google Cardboard

Table 1

7.2 Issues:

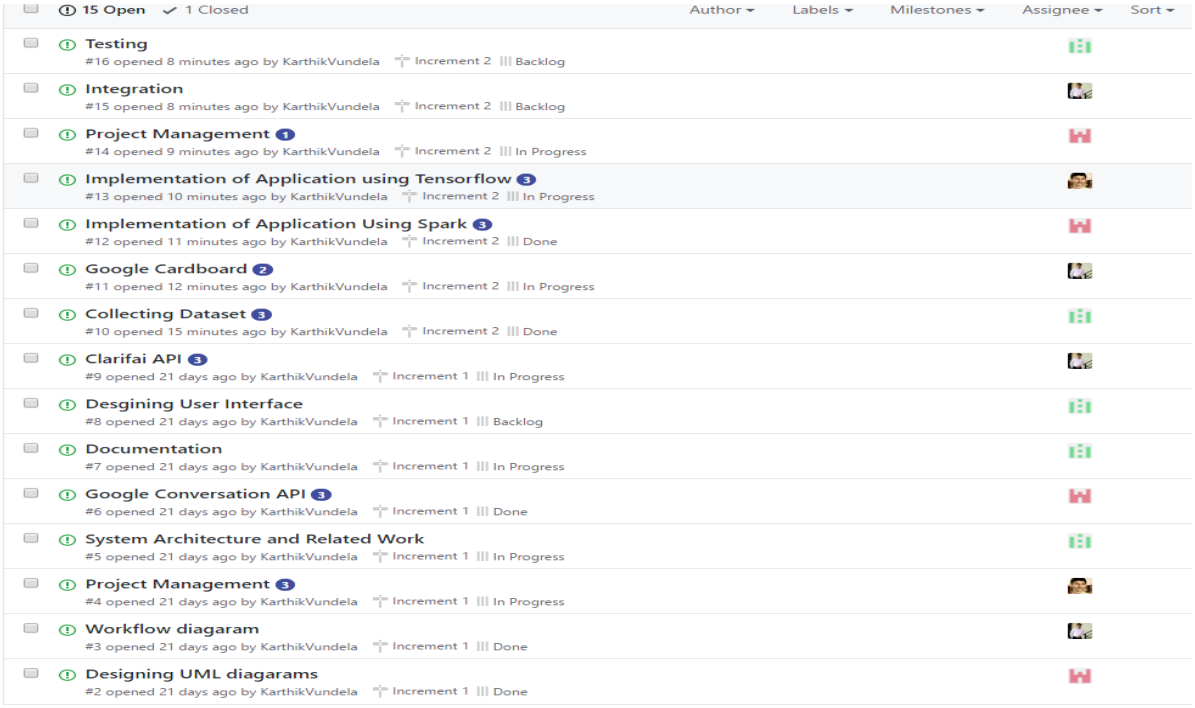


Fig 13

7.3 Burndown Charts:

7.3.1 Increment 1:

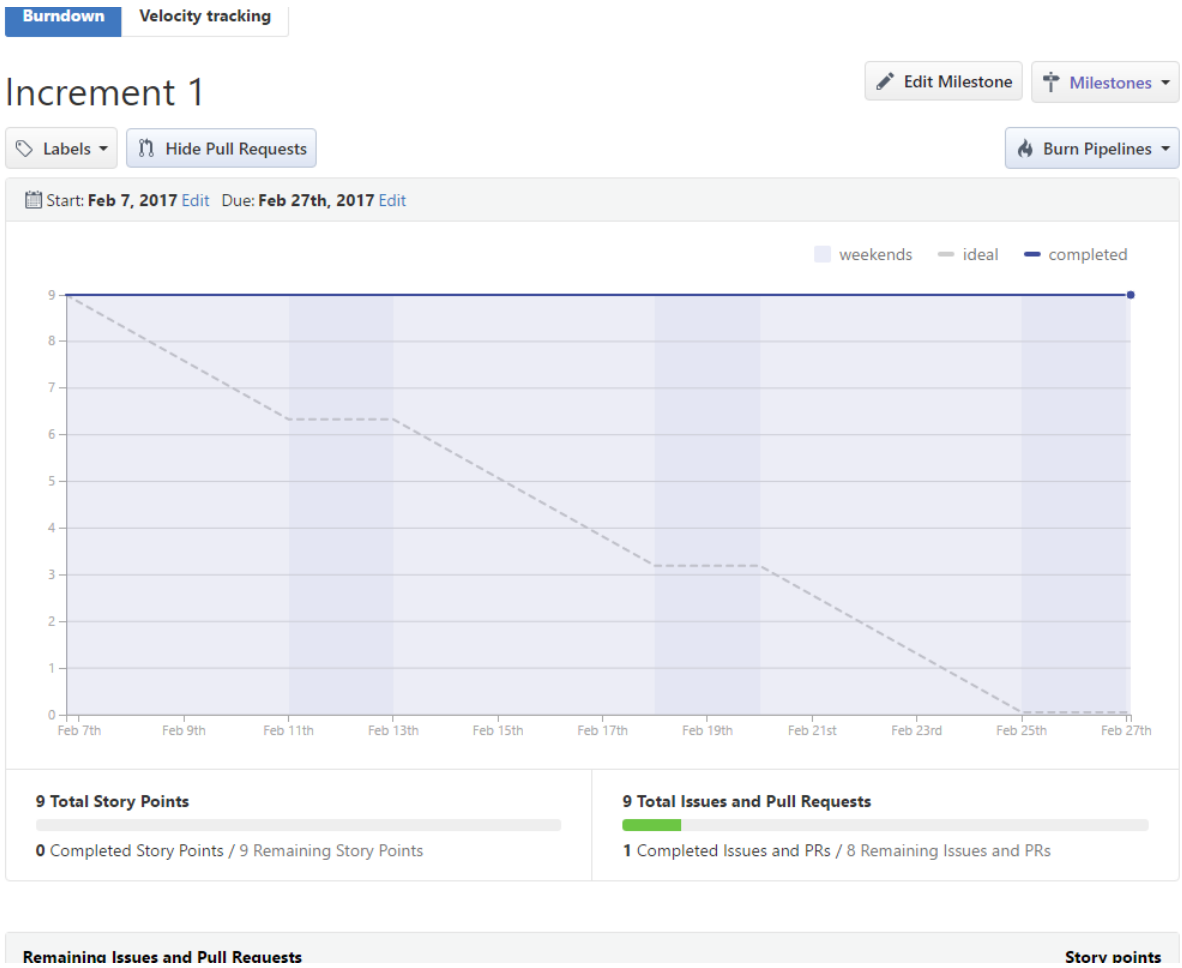


Fig 14

7.3.2 Increment 2:

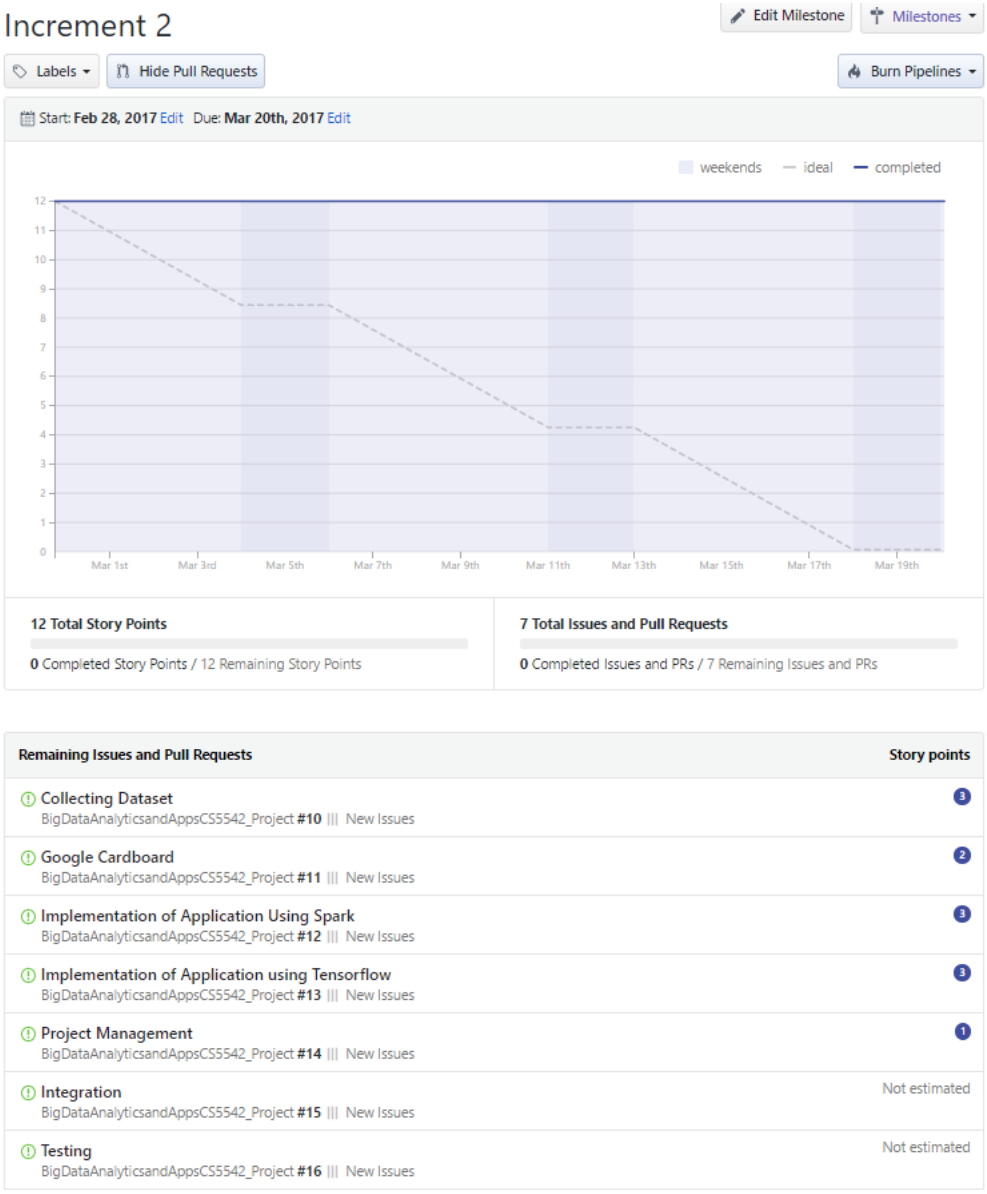


Fig 15

7.4 Zenhub Board:

7.4.1 After completion of Increment 1:

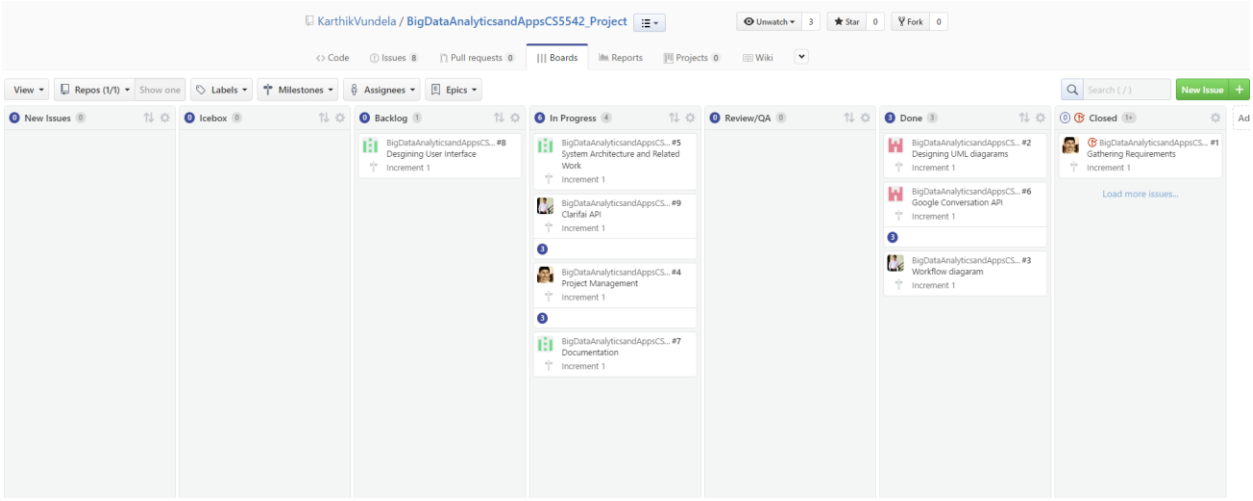


Fig 16

7.4.2 After completion of Increment 2:

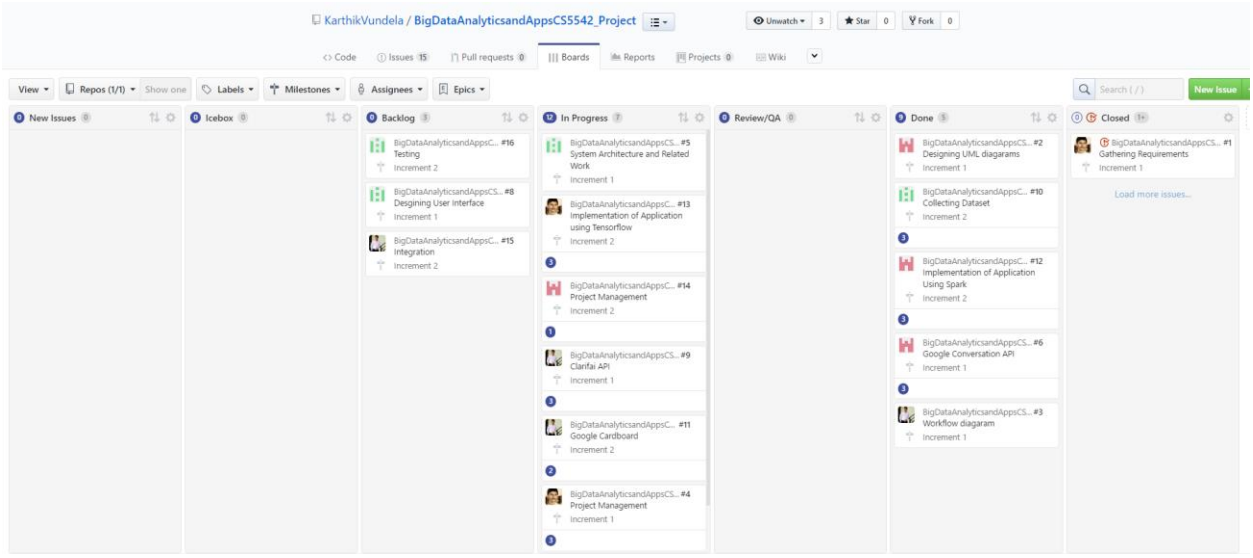


Fig 17

7.5 Work Completed:

Google Conversation, Clarifai API, Designing UML Diagarams, Software Architecture, Gathering Requirements, Data set collection, Spark Application, Tensor flow Application

7.6 Work need to be Completed:

Google Cardboard API

8 References:

Tutorial code 3,5

https://github.com/nteetor/Image-Classifier

http://www.di.ens.fr/willow/teaching/recvis09/final_project/