INTERNSHIP REPORT

PROJECT:

AnnoDB (Annotations Database)

Performed by:

Saurav Kumar Singh

1609033

Under the guidance of

Mrs. Mariya Moiz	Dr. S C Dutta
Sn. Principle Engineer	HOD (CS & IT)
Interra Systems, Noida	B.I.T. SINDRI
(Industry Mentor)	(Faculty Mentor)

Department of Computer Science & Engineering
Birsa Institute of Technology, Sindri
Dhanbad, Jharkhand - 828123
June 2018 - July 2018

DECLARATION

I hereby declare that the project work entitled "AnnoDB" is an authentic record of my own work carried out at Interra Systems, Noida under the guidance of Ms. Mariya Moiz (Industry Mentor) and Dr. S C Dutta (Faculty Mentor), during June 2018 to July 2018.

I also confirm that, the report is only made for my Academic requirement not for any other purpose.

(Signature of Student)
Saurav Kr. Singh
1609033

Date: ___________________

Certified that the above statement made by the student is correct to the best of our knowledge and belief.

Mrs. Mariya Moiz	Dr. S C Dutta
Sn. Principle Engineer	HOD (CS & IT)
Interra Systems, Noida	B.I.T., Sindri
(Industry Mentor)	(Faculty Mentor)

Experience Certificate from company / Certificate on Company Letterhead

ACKNOWLEDGEMENT

The satiation and euphoria that accompany the successful completion of the project would be incomplete without the mention of the people who made it possible

First and foremost I would like to thank Almighty god for giving me strength, health and ability, without which I could not have completed my Internship.

I would like to take the opportunity to thank and express my deep sense of gratitude to my corporate mentor Ms. Mariya Moiz and my faculty mentor Dr. S C Dutta. I am greatly indebted to both of them for providing their valuable guidance at all stages of the study, their advice, constructive suggestions, positive and supportive attitude and continuous encouragement, without which it would have not been possible to complete the project.

I would also like to thank Mr. Shailesh (my Company Manager) who in spite of busy schedule has cooperated with me continuously and indeed, his valuable contribution and guidance have been certainly indispensable for my project work. I would also like to thank him for giving me the opportunity to work with Interra Systems and learn.

I would also like to thank Dr. D K Singh (Director BIT Sindri), Dr. Ghanshyam Rai (Training & Placement Officer,BIT Sindri), Dr. S C Dutta (HOD, Department of CS/IT) and Prof. Raghunandan (Dept of IT, BIT Sindri), for giving me this novel chance to soak in the industrial experience and be adequately prepared for the rigors of the real world software development environment.

I also take this opportunity to thank my superiors at Interra Systems Mr. Anurag Upadhyay and Mr. Shashank Anand for their constant support, intelligent guidance and encouragement that helped me through various tasks and hurdles throughout the internship.

Finally, I must express my deepest gratitude to my parents and brothers for providing me with unfailing support and continuous encouragement throughout my time of study and through the process of researching and writing this report. This accomplishment would not have been possible without them.

I hope that I can build upon the experience and knowledge that I have gained and make a valuable contribution towards this industry in coming future.

Thank You! - Saurav Kumar Singh

CONTENTS

Chapter 1

1.1 Introduction

For Computer Science (CS) graduates, working at Interra Systems is nothing less than a beautiful dream. Not everyone gets a chance to work at Interra Systems, let alone without a graduation degree at hand. I wouldn't be wrong in saying that working at Interra Systems was one of the most thrilling and exhilarating experiences of my life.

Being a development engineer(Intern) in Team 'AnnoDB', Firstly I had to research on various development factors to understand and get a grip on the system of code already written. Later on my day to day tasks Included writing scripts in Python, handling PostgreSQL using Django ORM and working with REST APIs. Later I was given the task of Automated unit testing and performance testing. Further I had to generate HTML report of test cases and make performance graph of APIs.

Being at Interra Systems, the first thing I learned was the crucial importance of time. Everybody here is in a perpetual rush to meet deadlines. Each day you say, "It's still day one" and work with the same tempo. Everybody is just devising ways to save time by automating repetitive tasks that have a definite workflow. I had a splendid time in securing knowledge about how a service industry works, then the company's code of conduct, moral ethics and organizational hierarchy. Everybody here is supportive and interactive and they discuss various deep prospects of the tech industry and the phases of development in the project.

Working at such enormous and prodigious company, Interra Systems it was a privilege and I am always indebted to them for trusting me and providing me with such immense knowledge which helped to expand the boundaries of thinking and also providing me a great place to work.

1.2 About The Company

Interra Systems is a global provider of enterprise-class solutions that streamline the classification, quality control (QC) process, and monitoring of media content across the entire creation and distribution chain. Relying on Interra Systems' comprehensive video insights, media businesses can deliver video with high quality of experience, address new market trends, and improve monetization.

Widely adopted by broadcast, cable, telco, satellite, IPTV, OTT, and post-production markets around the world, Interra Systems' products enable better quality video, reduced exposure to regulatory issues, and higher customer satisfaction. Featuring AI- and machine learning-enabled algorithms, along with a flexible, software-defined architecture, Interra Systems' solutions support a variety of deployment scenarios, including the cloud, for higher performance, scalability, and efficiency.

The company's industry-leading solutions include BATON, a next-generation hybrid QC solution that delivers comprehensive capabilities way beyond standard automated QC; ORION and ORION OTT real-time content monitors assuring high QoE; and VEGA media analyzers for compliance and debug of encoded streams.

Solutions Provided:

BATON – Next Generation Hybrid QC for File-based Workflow

Interra Systems' BATON is the leading unified hybrid QC platform that implements organizational QC policy to support a combination of automated and manual QC checks – the result is a well integrated and efficient broadcast workflow. BATON is used by global broadcast, cable, telco, satellite, IPTV, over-the-top (OTT), and post-production markets and archiving companies working with file-based content.
BATON is the trusted choice for all their file-based QC needs with its comprehensive quality and ABR checks, scalability, and support for wide range of media formats, and an intuitive web-based interface.

ORION - Real-time Content Monitor

Interra Systems' ORION is a real-time content monitoring system that enables service providers to deliver error-free, superior quality video. ORION performs critical monitoring functions on hundreds of services simultaneously from a single platform, providing an operator with a single point of visibility and access to important information such as, status, alerts, alarms, visible impairments, error reports, triggered captures and more. ORION effortlessly monitors streaming content scaling single to multiple monitoring units/ sites across different geographical locations.

ORION-OTT - Content Monitor for OTT Workflow

Interra Systems' ORION™–OTT is a software-based over-the-top (OTT) monitoring solution for checking content integrity and related network performance of ABR content for multiscreen service delivery. Leveraging industry-proven audio/video quality analysis technologies, ORION OTT enables OTT service/equipment providers, content delivery networks (CDN) providers to seamlessly monitor online video delivery for quality and compliance.

VEGA – Media Analyzers for Standards Compliance, Debug, and Interoperability

VEGA is an industry leading media analysis platform for standards compliance, debug, and interoperability of encoded streams. VEGA enables navigation down to the deepest levels of a media file to generate error reports and analysis. This significantly reduces R&D and QA time in delivering standards-compliant video. VEGA supports all popular video compression and container standards and includes features such as video comparison and quality checks. These features help deliver high quality, standards-compliant media while reducing R&D and QA time.

Chapter - 2

2.1 Tools & Softwares used for Development

1. Visual Studio Code

Visual Studio Code is a source code editor developed by Microsoft for Windows, Linux and macOS. It includes support for debugging, embedded Git control, syntax highlighting, intelligent code completion, snippets, and code refactoring. It is also customizable, so users can change the editor's theme, keyboard shortcuts, and preferences. It is free and open-source, although the official download is under a proprietary license. It supports a number of programming languages and a set of features that may or may not be available for a given language.

2. Postman

Postman is a powerful HTTP client for testing web services. Postman makes it easy to test, develop and document APIs by allowing users to quickly put together both simple and complex HTTP requests. Postman is available as both a Google Chrome Packaged App and a Google Chrome inbrowser app. The packaged app version includes advanced features such as OAuth 2.0 support and bulk uploading/importing that are not available in the in-browser version. The in-browser version includes a few features, such as session cookies support, that are not yet available in the packaged app version.

3. pgAdmin

pgAdmin is the most popular and feature rich Open Source administration and development platform for PostgreSQL. There are native builds available for Linux, FreeBSD, Solaris, Mac OSX and Windows. Use it to manage PostgreSQL 7.3 and above running on any platform, as well as derived versions such as Postgres Plus Advanced Server and Greenplum database. The graphical interface covers anything from writing simple SQL queries to developing complex databases. A new major version is usually released in connection with every major version of PostgreSQL and supports all PostgreSQL features. Page | 11 It includes an SQL editor with syntax highlighting and a graphical query builder, an SQL/batch/shell job scheduling agent, support for Slony-I replication engine, a scripting engine and much more. Connect to any server via TCP/IP or via Unix Domain Sockets on *nix platforms, optionally SSL encrypted. pgAdmin is available in many languages. It is Free Software released under the PostgreSQL License

2.2 Technologies & Programming Languages used for Development

1. **Python**

   Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together.

2. **Django**

   Django is a high-level Python Web framework that encourages rapid development and clean, pragmatic design. Built by experienced developers, it takes care of much of the hassle of Web development, so you can focus on writing your app without needing to reinvent the wheel. It's free and open source.

3. **Java SE**

   Java Platform, Standard Edition (Java SE) lets you develop and deploy Java applications on desktops and servers. Java offers the rich user interface, performance, versatility, portability, and security that today's applications require.

4. **HTML**

   Hypertext Markup Language is the standard markup language for creating web pages and web applications. With Cascading Style Sheets and JavaScript, it forms a triad of cornerstone technologies for the World Wide Web.

5. **CSS**

   Cascading Style Sheets is a style sheet language used for describing the presentation of a document written in a markup language like HTML. CSS is a cornerstone technology of the World Wide Web, alongside HTML and JavaScript.

6. **Javascript**

   JavaScript, often abbreviated as JS, is a high-level, interpreted programming language. It is a language which is also characterized as dynamic, weakly typed, prototype-based and multi-paradigm.

Chapter - 3

Details of Work

3.1 Task 1

Script to delete trash marked items & delete their Annotations too.

```
from image.models import (Image,)
from manual_annotation.constants import (TRASH,)
from manual_annotation.models import (Annotation, EntityLifeCycle, Editor)
import json
def trashMarkedItems(self):
imageMap = {}
images = Image.objects.filter(tag=TRASH)
imageToBeDeleted=images.values_list('id',flat=True)
for image in images.iterator():
EntityLifeCycleDelete = EntityLifeCycle.objects.filter(image=image)
EditorDelete=Editor.objects.filter(lifecycle=lc) EditorDelete.delete() annotations =
Annotation.objects.filter(image_id=image)
imageMap[image.id]=annotations.values_list('id',flat=True) EditorDelete.delete()
EntityLifecycleDelete.delete()
annotations.delete()
filename= 'trashlog.json'
with open(filename,'w+') as outfile: json.dump(imageToBeDeleted,outfile)
image.delete()
```

3.2 Task 2

Script to move content where frame rate is greater than 24fps using Django ORM

```
import logging
import json
from django.core.management import BaseCommand
from django.core import serializers
from content.models import Content
from image.models import Image
from manual_annotation.models import Annotation

class Command(BaseCommand):

   def handle(self, *args, **options):
       contents = (Content.objects.all())
       content_idList = []
       image_id_List = []
       annotation_id_List = []
       for item in contents:
           try:
               obj = json.loads(item.content_info)
               if obj["frame_rate"] > "24":
                   image_id_List += self.get_imageids_from_sourceid(item.id)
                   content_idList.append(item.id)
           except KeyError:
               pass
       for id in image_id_List:
           annotation_id_List += self.get_annotationids_from_imageid(id)

   def get_imageids_from_sourceid(self, id):
       image_idsList = []
       image_ids = Image.objects.filter(source__id=id)
       for ids in image_ids:
           image_idsList.append(ids.id)
       return image_idsList
   def get_annotationids_from_imageid(self, id):
       Annotationids_List = []
       Annotation_ids = Annotation.objects.filter(image_id_id=id)
       for ids in Annotation_ids:
           Annotationids_List.append(ids.id)
       return Annotationids_List
```

3.3 Task 3

Script for vacuuming in PostgreSQL

```
import psycopg2
conn = psycopg2.connect
      (database = "annodb", user = "postgres", password =
       "*********", host = "localhost", port = "5432")
      print ("Opened database successfully")
cur = conn.cursor()
conn.autocommit = True cur.execute("VACUUM ")
conn.autocommit = False
print("vacuuming done !")
conn.close()
```

3.4 Task 4

Create a demo database and dump database contents as JSON

```
#Create Operation

import psycopg2
conn = psycopg2.connect(database = "annodb", user = "postgres", password =
"*******", host = "localhost", port = "5432")
print("Opened database successfully")

cur = conn.cursor()
cur.execute('''CREATE TABLE IMAGE_PATH
      (ID INT PRIMARY KEY     NOT NULL,
      PATH           TEXT    NOT NULL);''')
print("Table created successfully")
conn.commit()
conn.close()
```

```
#Insert Operation
import psycopg2

conn = psycopg2.connect(database = "annodb", user = "postgres", password =
"********", host = "localhost", port = "5432")
print("Opened database successfully")
cur = conn.cursor()
paths=[]
ids=[]
ids.append("101")
paths.append(r"C:\Users\8ssingh\Desktop\src\download.jpg")
ids.append("102")
paths.append(r"C:\Users\8ssingh\Desktop\src\download(1).jpg")
ids.append("103")
paths.append(r"C:\Users\8ssingh\Desktop\src\download(2).jpg")
ids.append("104")
paths.append(r"C:\Users\8ssingh\Desktop\src\download(3).jpg")

for i in range(len(ids)):
    cur.execute("INSERT INTO IMAGE_PATH (ID,PATH) VALUES (%s,
%s)",(ids[i],paths[i]));
print('Elements inserted successfully')
conn.commit()
conn.close()
```

```
#Dump as JSON

import psycopg2
import json

conn = psycopg2.connect(database = "annodb", user = "postgres", password =
"********", host = "localhost", port = "5432")
print ("Opened database successfully")

cur = conn.cursor()
querry="SELECT * FROM IMAGE_PATH"
cur.execute(querry);
result=cur.fetchall()
print(result)
items= []
print(type(result))
for row in result:
    items.append({"id":row[0], "path":row[1]})
print ("Operation done successfully")
conn.commit()
conn.close()
jsonData= json.dumps({"IMAGE DETAILS": items})
print(jsonData)
```

3.5 Task 5

Move file from source to destination using Django ORM and handle every possible exceptions

```python
from content.models import Content
from image.models import Image
import os
class ChangeDestination:

   def path_exists(self, path):
       return os.path.exists(path)

   def valid_dir(self, path):
       return os.path.isdir(path)

   def file_exists(self, path):
       return os.path.isfile(path)

   def get_or_create_path(self, path, recursive=True):
       if not os.path.exists(path):
           if recursive:
               os.makedirs(path)
           else:
               os.mkdir(path)

   def rename_file(self,src,index):
       index+=1
       filepath, filename = os.path.split(src)
       extension = os.path.splitext(filename)
       extensionlist=list(extension)
       extensionlist[0]=extension[0]+str(index)
       filname="".join(extensionlist)
       return os.path.join(filepath,filname)

   def change_path(self,src,dst):
       if self.path_exists(src):
           self.get_or_create_path(dst)
       else:
           print("file does not exists try again \n "+ src)
           return None
       temp=src
       filepath, filename = os.path.split(src)
       src1=os.path.join(dst,filename)
       src=src1
       index=1
```

```
        while(os.path.exists(src1)):
            src1=self.rename_file(src,index)
            index = index+1
        print(" files are successfully moved to",src1)
        os.rename(temp, src1)
        print("file moved successfully ")
        return src1

    def move_file(self,dst):
        temp1 = list(Image.objects.values('path','name','source__path'))
        for i in range(len(temp1)):
            newLoc= self.change_path(temp1[i]['path'],dst)
            var =
Image.objects.filter(path=temp1[i]['path']).update(path=newLoc)
            var.save()
```

3.6 Task 6

Perform HttpGET request using Java and print its Response, Response code and Headers.

```
import java.io.IOException;
import java.util.Base64;
import java.util.HashMap;
import org.apache.http.Header;
import org.apache.http.HttpHeaders;
import org.apache.http.HttpResponse;
import org.apache.http.client.ClientProtocolException;
import org.apache.http.client.HttpClient;
import org.apache.http.client.methods.HttpGet;
import org.apache.http.impl.client.HttpClientBuilder;
import org.apache.http.util.EntityUtils;
import org.json.JSONException;
import org.json.JSONObject;

public class Executor {
     public void get(String url) throws ClientProtocolException, IOException,
JSONException
     {
          //Status code
          HttpGet httpGet= new HttpGet(url);
          String auth = "admin" + ":" + "********";
          String encodedAuth =
Base64.getEncoder().encodeToString(auth.getBytes());
//.encodeBase64(auth.getBytes(Charset.forName("US-ASCII")));
          String authHeader = "Basic " + new String(encodedAuth);
          System.out.println(authHeader);
          httpGet.setHeader(HttpHeaders.AUTHORIZATION, authHeader);
          HttpClient httpClient = HttpClientBuilder.create().build();
          HttpResponse httpResponse = httpClient.execute(httpGet);
          httpResponse = httpClient.execute(httpGet);
          System.out.println("Response code is " +
httpResponse.getStatusLine().getStatusCode());
          //Json String
          String responseString =
EntityUtils.toString(httpResponse.getEntity(),"UTF-8");
          //System.out.println(responseString);
          JSONObject responseJSON=new JSONObject(responseString);
          System.out.println("json output"+ responseJSON);
          //All Headers
          Header[] headerArray = httpResponse.getAllHeaders();
          HashMap < String, String > allHeaders = new
HashMap<String,String>();
```

```
            for(Header header:headerArray)
            {
                  allHeaders.put(header.getName(), header.getValue());
            }
            System.out.println("headers Array " + allHeaders );
      }
      public static void main(String[] args) {
            // TODO Auto-generated method stub

      }

}
```

3.7 Task 7

Perform HttpGET request of all REST API’s using Python and print their response time.

```
import requests
from requests.auth import HTTPBasicAuth
import time
millisrequest= round(time.time(),2)
url="http://192.168.4.72:8000/rest/accounts/daily-user-summary.json"
r = requests.get(url,auth=HTTPBasicAuth('admin','********'))
millisresponse = round(time.time(),2)
print("Response time of API -  rest/content/content-summary.json  is = " ,
round(millisresponse-millisrequest,2), "s")
```

3.8 Task 8

Functional Testing

- Get the list of all users
- One by one call post to login
- Response will contains user details and permissions
- Check for 'content.restricted' permission in the permission lists
- If user has restricted permission, he should get the restricted labels as well in 'allLabels.json'
- If not, no restricted labels should be present

Perform this test for both Explicit Label access and Explicit Content Access

```
import requests
from requests.auth import HTTPBasicAuth
import unittest
import traceback
import time
import base64
import allure
import pytest

class TestCase_accounts(unittest.TestCase):
    def setUp(self):
        self.url ="http://192.168.4.72:8000/rest/accounts/users"
        with allure.step('GET API : ' + self.url+ "\n"  ):
            self.response1 =
requests.get(self.url,auth=HTTPBasicAuth('admin','********'))
            try:
                with allure.step("  HTTP Response  --  " + str(self.response1)):
                    assert self.response1.status_code==200
            except:
                raise Exception("HTTP Response Error " +
str(self.response1.status_code))
            self.jsnres=self.response1.json()
        """list of all users"""
     @allure.step('Test Complete')
    def tearDown(self):
       with allure.step('Testing of API : ' + self.url+ "\n" + "    Complete"
):
            pass

    @allure.feature("test case")
    def test_explicit_action_label_access(self):
        info=""
        try:
            print("testing explicit action label access")
            username= []
            password=[]
            for i in range(len(self.jsnres)):
                temp=self.jsnres[i]["username"]
username.append(base64.b64encode(temp.encode('utf-8')).decode("utf-8")  )
password.append(base64.b64encode("*********".encode('utf-8')).decode("utf-8"))
                with allure.step(" POST REQUEST :
http://192.168.4.72:8000/rest/accounts/login"):
                   response2=requests.post(url =
"http://192.168.4.72:8000/rest/accounts/login", data =
{'username':username[i],'password':password[i]})
                   with allure.step("POST Data --- username " + username[i] + "
-- password --" + password[i] ):
                       try:
                           with allure.step("  HTTP Response  --   " +
str(response2)  ):
                               assert response2.status_code==200
                       except:
                           raise Exception("HTTP Response Error " +
str(response2.status_code))
                response3=response2.json()
                info = str(response3)
                if 'content.restricted' in response3["user"]["permissions"]:
                    response4=
requests.get("http://192.168.4.72:8000/rest/label/allLabels.json",auth=HTTPBasi
cAuth(temp,'xp#$kal1'))
                    try:
                        assert response4.status_code==200
                    except:
                        info= ""
                        raise Exception("HTTP Response Error " +
str(response4.status_code))
                    response5=response4.json()
                    info = str(response5)

                    self.assertTrue("Explicit Action" in response5, True)
                else:
                    response4=
requests.get(url="http://192.168.4.72:8000/rest/label/allLabels.json",auth=HTTP
BasicAuth(temp,'*******'))
                    try:
                        assert response4.status_code==200
                    except:
                        raise Exception("HTTP Response Error " +
str(response4.status_code))
```

```
                    response5=response4.json()
                    info = str(response5)
                    self.assertFalse("Explicit Action" in response5, True)
        except AssertionError:
            self.fail("AssertionError")
        except:
            tb = traceback.format_exc()
            raise Exception("\n Some Error in the system : \n" + tb + " \n
----- Info received from the server -------- \n  " + info )

    @allure.feature("test case")
    def test_explicit_content_access(self):
        try:
            print("testing explicit content access")
            username= []
            password=[]
            failUsers=[]
            for i in range(len(self.jsnres)):
                temp=self.jsnres[i]["username"]

username.append(base64.b64encode(temp.encode('utf-8')).decode("utf-8")  )
password.append(base64.b64encode("********".encode('utf-8')).decode("utf-8"))
                with allure.step(" POST REQUEST :
http://192.168.4.72:8000/rest/accounts/login"):
                   response2=requests.post(url =
"http://192.168.4.72:8000/rest/accounts/login", data =
{'username':username[i],'password':password[i]})
                   with allure.step("POST Data --- username " + username[i] + "
-- password --" + password[i] ):
                       try:
                           with allure.step("  HTTP Response  --   " +
str(response2)  ):
                               assert response2.status_code==200
                       except:
                           raise Exception("HTTP Response Error " +
str(response2.status_code))
                response3=response2.json()
                info = str(response3)
                """ Enter the id to be checked"""
                payload = {'id': 43}
                with allure.step("Content Restricted in user Permission"):
                   if 'content.restricted' in response3["user"]["permissions"]:
                       with allure.step("GET REQUEST : " +
"http://192.168.4.72:8000/rest/image/imagedetails.json" + str(payload)):
                           response4=
requests.get(url="http://192.168.4.72:8000/rest/image/imagedetails.json",params
=payload,auth=HTTPBasicAuth(temp,'********'))
                           try:
                               with allure.step("HTTP Response -- " +
str(response4)):
                                   assert response4.status_code==200
                           except:
                               raise Exception("HTTP Response Error " +
str(response4.status_code))
                       response5=response4.json()
                       info = str(response5)
                       if ("Restricted" in response5["access"] or "R" in
response5["access"] or "General" in response5["access"] or "G" in
response5["access"])!=True:
                           print("r")
                           failUsers.append(temp)
                   else:
                       with allure.step("GET REQUEST : " +
"http://192.168.4.72:8000/rest/image/imagedetails.json" + str(payload)):
                           response4=
requests.get(url="http://192.168.4.72:8000/rest/image/imagedetails.json",params
=payload,auth=HTTPBasicAuth(temp,'********'))
                           try:
                               with allure.step("HTTP Response -- " +
str(response4)):
                                   assert response4.status_code==200
                           except:
                               raise Exception("HTTP Response Error " +
str(response4.status_code))
                       response5=response4.json()
                       info = str(response5)
                       try:
                           if "Access denied" in response5["detail"]!=True:
                               print('g')
                               failUsers.append(temp)
                           elif "General" in response5["access"] or "G" in
response5["access"] !=True:
                               print("g")
                               failUsers.append(temp)
                       except KeyError:
                           pass
            print("List of fail cases(users)",failUsers)
            self.assertEqual(len(failUsers),0, "list of fail users" +
str(failUsers))
        except AssertionError:
            self.fail("Assertion error    "  + " list of failed users    : " +
str(failUsers) )
```

```
        except:
            tb = traceback.format_exc()
            raise Exception("\n Some Error in the system : \n" + tb + " \n
----- Info received from the server -------- \n  " + info )
```

3.9 Task 9

<u>Functional Testing</u>

Testing Status of task id’s

Testing whether scheduled time is greater than start time

Altogether 21 functional testing test cases

```
import requests
from requests.auth import HTTPBasicAuth
import unittest
import time
import HtmlTestRunner
import traceback
import allure
import pytest

class TestCase_worker(unittest.TestCase):
   def setUp (self):

       self.url ="http://192.168.4.72:8000/rest/worker/tasklist.json"
       with allure.step('GET API : ' + self.url+ "\n"    ):
           self.r = requests.get(self.url,auth=HTTPBasicAuth('admin','******'))
           try:
               with allure.step("  HTTP Response  --   " + str(self.r)  ):
                   assert self.r.status_code==200
           except:
               raise Exception("HTTP Response Error " +
str(self.r.status_code))
       self.res=self.r.json()

   @allure.step('Test Complete')
   def tearDown(self):
       with allure.step('Testing of API : ' + self.url+ "\n" + "    Complete"
):
           pass
   @allure.feature("test case")
   def test_Testing_of_Status(self):
       """Check status of task id's"""
       with allure.step('API Response'  ):

           try:
               print("\nTesting_of_Status\n")
               task_ids=[]
               info = ""
               for i in range(len(self.res['data'])):
                   with allure.step('checking status of task id = ' +
self.res['data'][i]["task_id"]+ ' --  Received Response : ' +
self.res['data'][i]["status"] + '       Expected  : not Failed'):
                       info = self.res['data'][i]
                       x = self.res['data'][i]["status"]!= "failed"
                   if x== False:
                       task_ids.append(self.res['data'][i]["task_id"])
               if len(task_ids)>0:
                   self.assertEqual(False, True)
               else:
                   self.assertEqual(True, True)
           except AssertionError:
               self.fail("\n task_ids in which error's were found are :  \n" +
' , '.join(task_ids))
           except:
               tb = traceback.format_exc()
           raise Exception("\n Some Error in the system : \n" + tb + " \n -----
Info received from the server -------- \n  " + str(info) )

   @allure.feature("test case")
   def test_Testing_of_Schedule_time(self):
       """Check whether shedule time is greter than start time"""
       with allure.step('API Response'  ):
…………………………………………………………………………….
…………………………………………………………………………….
…………………………………………………………………………….
…………………………………………………………………………….
```

3.10 Task 10

<u>Performance testing of ‘AnnoDB’</u>

Test Average response time of Annotation Export , Content / Image Export and Label Export under various parameters

Altogether 158 test cases

```
import requests
from requests.auth import HTTPBasicAuth
import unittest
import time
import traceback
import allure
import pytest

class Test_contentInfo(unittest.TestCase):
   def convertMillis(self,millis):
       millis =float (millis)
       ms=millis%1000
       seconds=(millis/1000)%60
       minutes=(millis/(1000*60))%60
       minutes = int(minutes)
       hours=(millis/(1000*60*60))%24
       return (hours, minutes, seconds,ms)

   @allure.feature('Response Time')
   @allure.story('Content/Image Export (Content Info)')
   def test_ContentInfo1(self):
       """API  --- /rest/image/images  \n
           Params = {access_type = Restricted ,
                     Status = Annotated
                     Annotation Type = image
                     Content Tag = None  } """
       APItime = []
       for i in range(5):
           url =
"http://192.168.4.72:8000/rest/content/contents?access_type=R&status=annotated&
entity_type=['image']&fields=id,name,path"
           millisrequest = int(round(time.time() * 1000))
           r = requests.get(url,auth=HTTPBasicAuth('admin','*********'))
           millisresponse = int(round(time.time() * 1000))
           APItime.append(millisresponse-millisrequest)
       responseTime = sum(APItime) / float(len(APItime))
       with allure.step("Avg Response Time ---- "  + "%dh:%dm:%ds:%dms" %
(self.convertMillis(responseTime))):
           pass
```

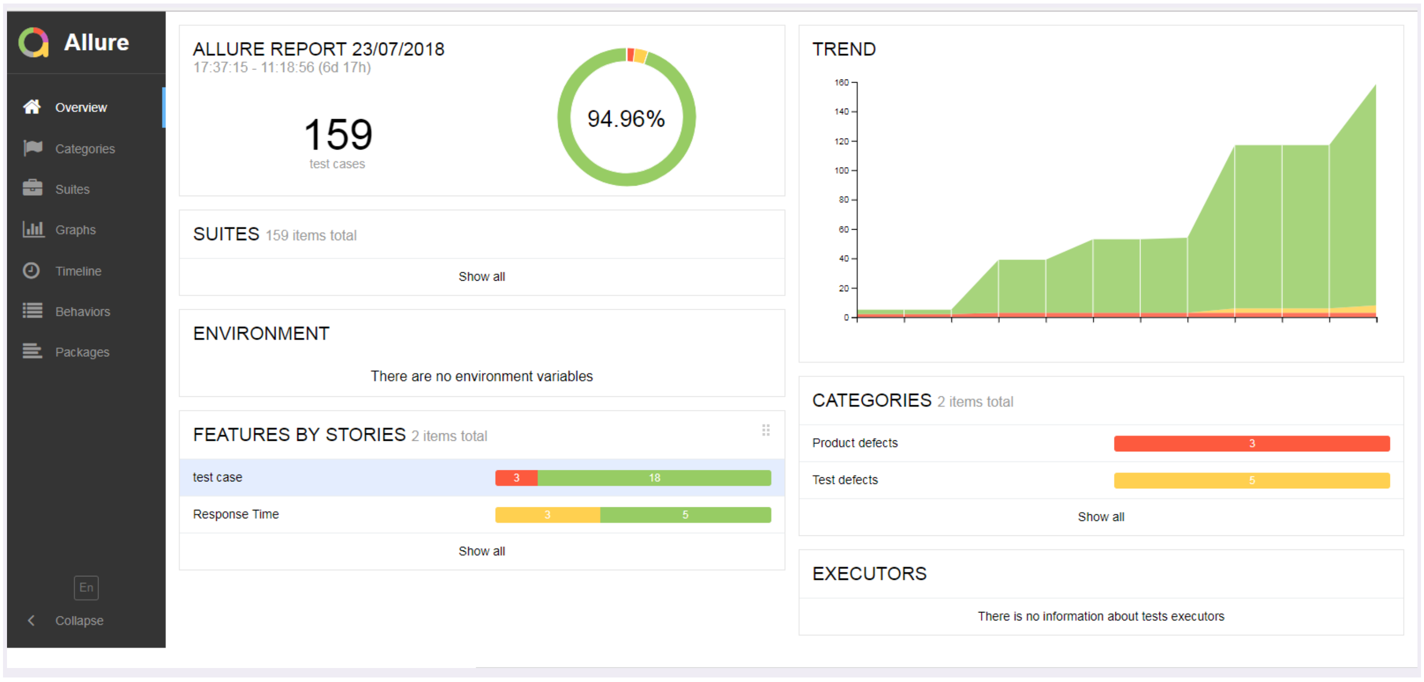
3.11 Task 11

Generate HTML Report of Test cases using **Allure 2.x**
Add description to all test cases
Add steps to all test cases
Categorise each test cases according to their feature and story.

Implementation:
From allure Documentation

```
Scoop install allure
Py.test --alluredir <report folder>(.report)
Allure generate -c report
Allure open allure-report --port 1314
```

```
@allure.feature("Feature")
@allure.Step("Step")
@allure.description("Description")
@allure.description("Story")
```



ALLURE REPORT 23/07/2018
17:37:15 - 11:18:56 (6d 17h)
159
test cases

SUITES 159 items total

ENVIRONMENT
There are no environment variables

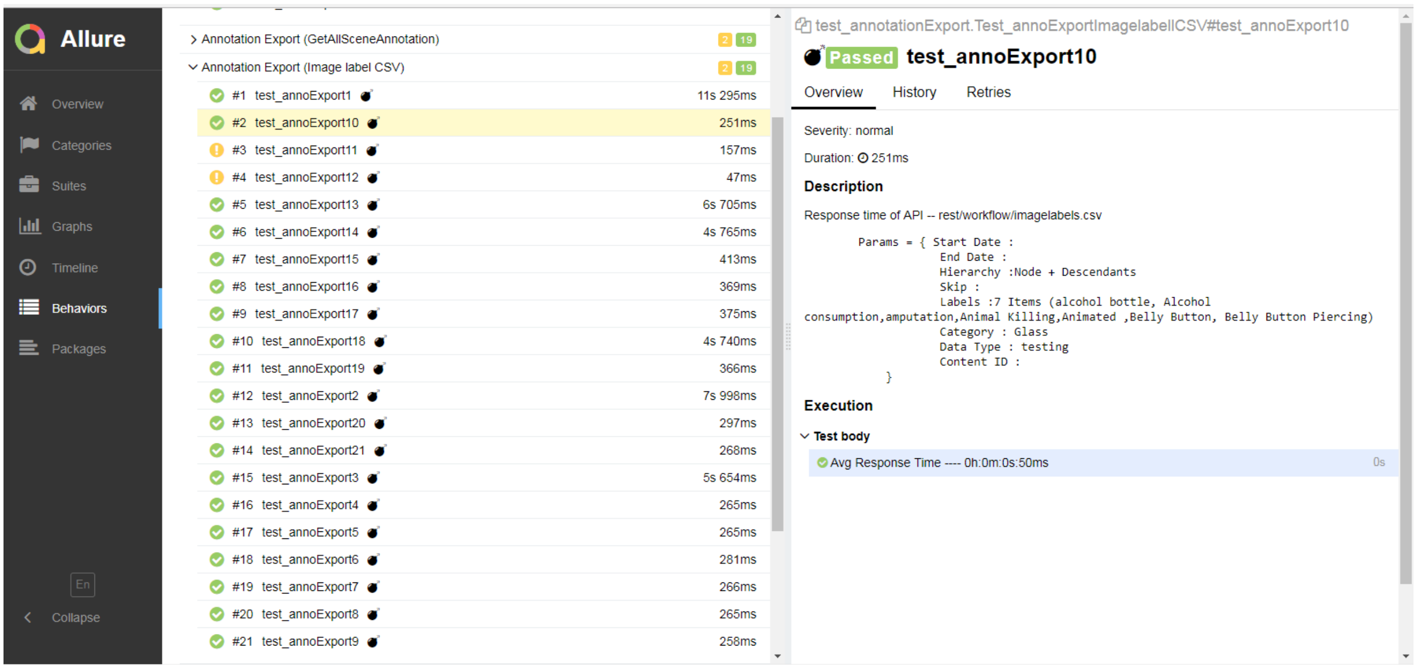
FEATURES BY STORIES 2 items total

test case	3	18
Response Time	3	5

CATEGORIES 2 items total

Product defects	3
Test defects	5

EXECUTORS
There is no information about tests executors


Allure

Overview
Categories
Suites
Graphs
Timeline
Behaviors
Packages
En
Collapse

```
> Annotation Export (GetAllSceneAnnotation)        2 19
v Annotation Export (Image label CSV)              2 19
  #1  test_annoExport1                          11s 295ms
  #2  test_annoExport10                            251ms
  #3  test_annoExport11                            157ms
  #4  test_annoExport12                             47ms
  #5  test_annoExport13                          6s 705ms
  #6  test_annoExport14                          4s 765ms
  #7  test_annoExport15                            413ms
  #8  test_annoExport16                            369ms
  #9  test_annoExport17                            375ms
  #10 test_annoExport18                          4s 740ms
  #11 test_annoExport19                            366ms
  #12 test_annoExport2                           7s 998ms
  #13 test_annoExport20                            297ms
  #14 test_annoExport21                            268ms
  #15 test_annoExport3                           5s 654ms
  #16 test_annoExport4                             265ms
  #17 test_annoExport5                             265ms
  #18 test_annoExport6                             281ms
  #19 test_annoExport7                             266ms
  #20 test_annoExport8                             265ms
  #21 test_annoExport9                             258ms
```

test_annotationExport.Test_annoExportImagelabelICSV#test_annoExport10

Passed **test_annoExport10**

Overview History Retries

Severity: normal

Duration: 251ms

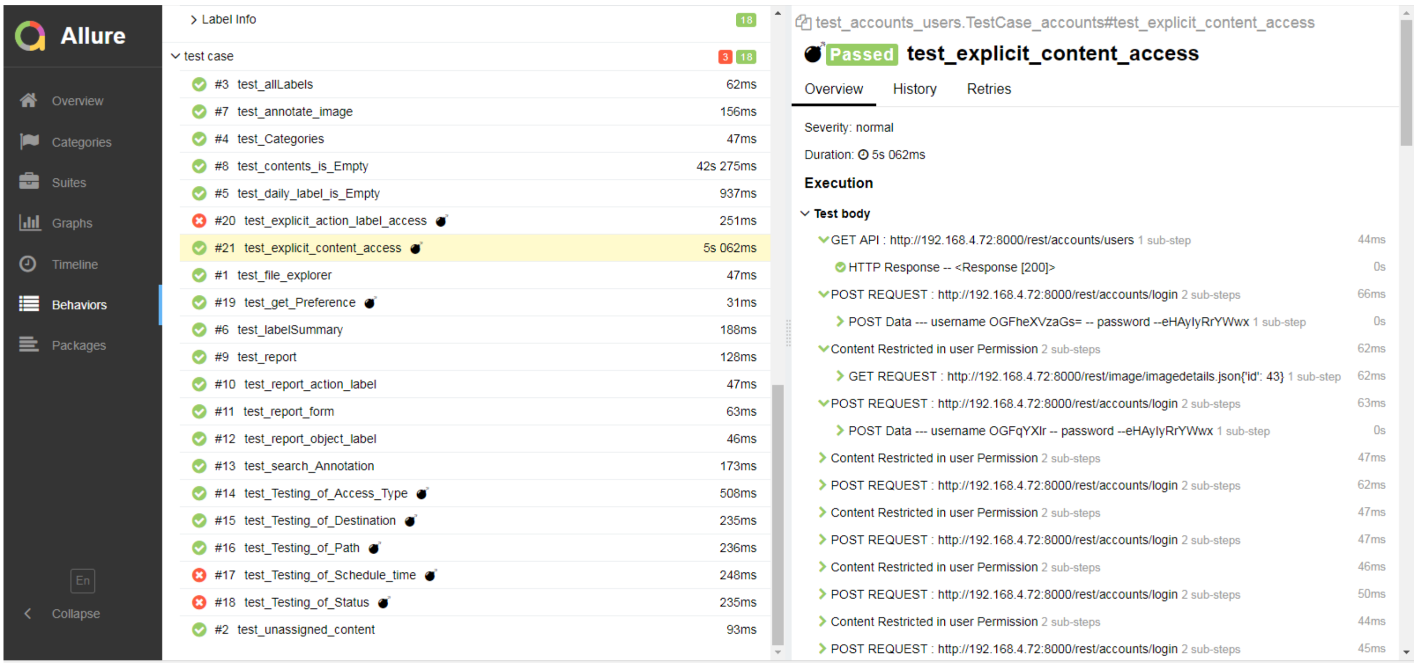
Description

Response time of API -- rest/workflow/imagelabels.csv

```
        Params = { Start Date :
                   End Date :
                   Hierarchy :Node + Descendants
                   Skip :
                   Labels :7 Items (alcohol bottle, Alcohol
consumption,amputation,Animal Killing,Animated ,Belly Button, Belly Button Piercing)
                   Category : Glass
                   Data Type : testing
                   Content ID :
           }
```

Execution

```
v Test body
  Avg Response Time ---- 0h:0m:0s:50ms                                0s
```


Allure

Overview
Categories
Suites
Graphs
Timeline
Behaviors
Packages
En
Collapse

```
> Label Info                                        18
v test case                                       3 18
  #3  test_allLabels                                62ms
  #7  test_annotate_image                          156ms
  #4  test_Categories                               47ms
  #8  test_contents_is_Empty                    42s 275ms
  #5  test_daily_label_is_Empty                    937ms
  #20 test_explicit_action_label_access            251ms
  #21 test_explicit_content_access              5s 062ms
  #1  test_file_explorer                            47ms
  #19 test_get_Preference                           31ms
  #6  test_labelSummary                            188ms
  #9  test_report                                  128ms
  #10 test_report_action_label                      47ms
  #11 test_report_form                              63ms
  #12 test_report_object_label                      46ms
  #13 test_search_Annotation                       173ms
  #14 test_Testing_of_Access_Type                  508ms
  #15 test_Testing_of_Destination                  235ms
  #16 test_Testing_of_Path                         236ms
  #17 test_Testing_of_Schedule_time                248ms
  #18 test_Testing_of_Status                       235ms
  #2  test_unassigned_content                       93ms
```

test_accounts_users.TestCase_accounts#test_explicit_content_access

Passed **test_explicit_content_access**

Overview History Retries

Severity: normal

Duration: 5s 062ms

Execution

```
v Test body
  v GET API : http://192.168.4.72:8000/rest/accounts/users 1 sub-step                        44ms
      HTTP Response -- <Response [200]>                                                       0s
  v POST REQUEST : http://192.168.4.72:8000/rest/accounts/login 2 sub-steps                  66ms
    > POST Data --- username OGFheXVzaGs= -- password --eHAyIyRrYWwx 1 sub-step               0s
  v Content Restricted in user Permission 2 sub-steps                                        62ms
    > GET REQUEST : http://192.168.4.72:8000/rest/image/imagedetails.json{'id': 43} 1 sub-step 62ms
  v POST REQUEST : http://192.168.4.72:8000/rest/accounts/login 2 sub-steps                  63ms
    > POST Data --- username OGFqYXlr -- password --eHAyIyRrYWwx 1 sub-step                   0s
  > Content Restricted in user Permission 2 sub-steps                                        47ms
  > POST REQUEST : http://192.168.4.72:8000/rest/accounts/login 2 sub-steps                  62ms
  > Content Restricted in user Permission 2 sub-steps                                        47ms
  > POST REQUEST : http://192.168.4.72:8000/rest/accounts/login 2 sub-steps                  47ms
  > Content Restricted in user Permission 2 sub-steps                                        46ms
  > POST REQUEST : http://192.168.4.72:8000/rest/accounts/login 2 sub-steps                  50ms
  > Content Restricted in user Permission 2 sub-steps                                        44ms
  > POST REQUEST : http://192.168.4.72:8000/rest/accounts/login 2 sub-steps                  45ms
```

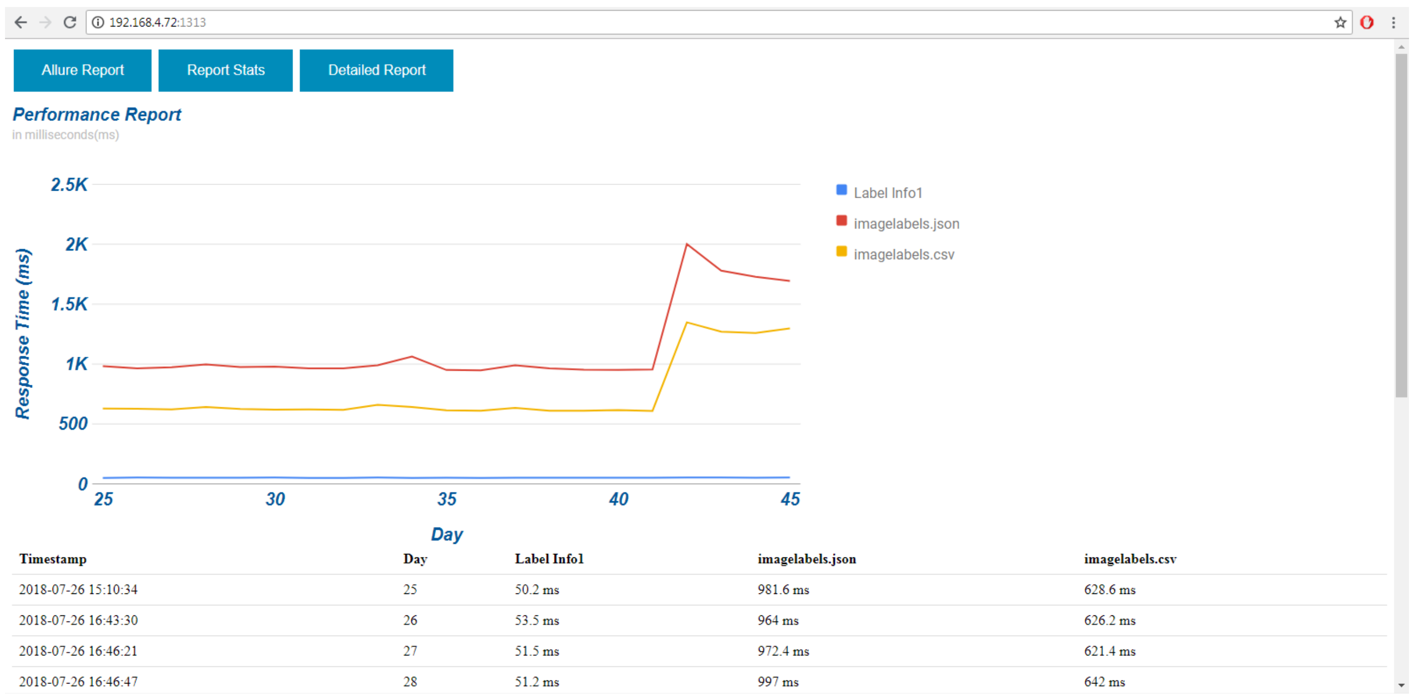
3.12 Task 12

Make Automated performance graph of 3 API’s.
Update this Graph daily.
Create a python script for this and add it to windows Task Scheduler.

Implementation:

Graph was created using Google Line charts. It’s daily Report data was appended in a .txt file in JSON format and using JavaScript, data for Graph and Table was fetched. For all these operation a python script was used.

Performance Graph was accessed using http-server (cors).



Timestamp	Day	Label Info1	imagelabels.json	imagelabels.csv
2018-07-26 15:10:34	25	50.2 ms	981.6 ms	628.6 ms
2018-07-26 16:43:30	26	53.5 ms	964 ms	626.2 ms
2018-07-26 16:46:21	27	51.5 ms	972.4 ms	621.4 ms
2018-07-26 16:46:47	28	51.2 ms	997 ms	642 ms

JS Script

```
<script type="text/javascript">
 function CreateTableFromJSON() {
   google.charts.load('current', {'packages':['line']});
   google.charts.setOnLoadCallback(drawChart);
   var local_data = data;

   obj=JSON.stringify(local_data[0])
       // CREATED DYNAMIC TABLE.
       var table = document.createElement("table");
       // CREATED HTML TABLE HEADER ROW USING THE EXTRACTED HEADERS ABOVE.
       var tr = table.insertRow(-1);                   // TABLE ROW.
       var th = document.createElement("th");      // TABLE HEADER.
           th.innerHTML = "Timestamp";
           tr.appendChild(th);
       for (var i = 0; i < local_data[0].cols.length; i++) {
           var th = document.createElement("th");      // TABLE HEADER.
           th.innerHTML = local_data[0].cols[i].label;
           tr.appendChild(th);
       }
       // ADDED JSON DATA TO THE TABLE AS ROWS.
       for (var i = 0; i < local_data[0].rows.length; i++) {
           tr = table.insertRow(-1);
           for (var j = -1; j < local_data[0].cols.length; j++) {
               var tabCell = tr.insertCell(-1);
               if (j == -1) {
                 tabCell.innerHTML = local_data[0].timestamp[i];
             }else if (j==0){
               tabCell.innerHTML = local_data[0].rows[i].c[j].v ;
             }
             else {
               tabCell.innerHTML = local_data[0].rows[i].c[j].v + " ms";
             }

           }
       }
       // FINALLY ADDED THE NEWLY CREATED TABLE WITH JSON DATA TO A CONTAINER.
       var divContainer = document.getElementById("showData");
       divContainer.innerHTML = "";
       divContainer.appendChild(table);
   }
```

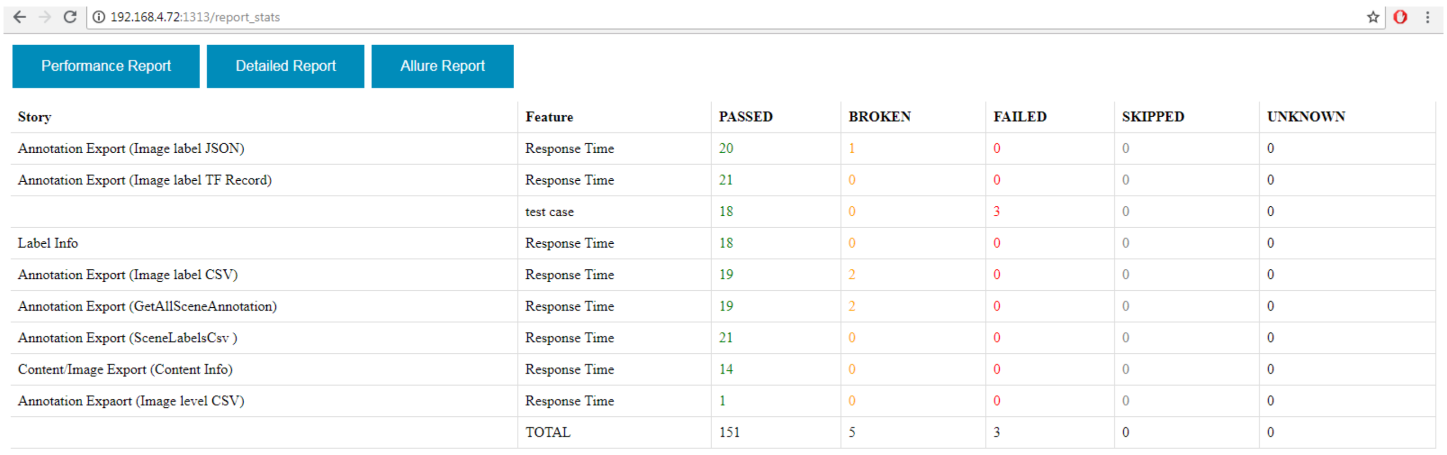
Python Script

```
class performancereport(unittest.TestCase):
    daydata = []
    @classmethod
    def tearDownClass(self):
        self.trendupdatedata(self.daydata)
    def trendupdatedata(self,daylist = daydata):
        file1 = open(r"C:\Users\8ssingh\Desktop\Google Charts\New 
folder/charts.txt","r+")
        data = file1.read()
        chartdata= (data.split('='))
        jsondata=json.loads(chartdata[1])
        daylist[0]=dict(v=daylist[0])
        daylist[1]=dict(v=daylist[1])
        daylist[2]=dict(v=daylist[2])
        if len(jsondata[0]["rows"])>20:
            timelist = deque(jsondata[0]["timestamp"])
            temp = timelist.popleft()
            jsondata[0]["timestamp_old"].append(temp)
            jsondata[0]["timestamp"]= list(timelist)
            jsondata[0]["timestamp"].append(datetime.now().strftime('%Y-%m-%d 
%H:%M:%S'))
            rot=deque(jsondata[0]["rows"])
            jsondata[0]["rows_old"].append((rot.popleft()))
            jsondata[0]["rows"]=list(rot)
            daylist.insert(0,dict(v=jsondata[0]["rows"][19]["c"][0]["v"]+1))
        else:
            jsondata[0]["timestamp"].append(datetime.now().strftime('%Y-%m-%d 
%H:%M:%S'))
            temp= len(jsondata[0]["rows"])-1

daylist.insert(0,dict(v=jsondata[0]["rows"][temp]["c"][0]["v"]+1))
        colsnew = dict(c= daylist)
        jsondata[0]["rows"].append(colsnew)
        file2 =open(r"C:\Users\8ssingh\Desktop\Google Charts\New 
folder/charts.txt","w")
        file2.write("data ="+ (json.dumps(jsondata)))
        file2.close();
        file1.close();

    def test_a_labelInfo1(self):
.
.
.
```

3.13 Task 13

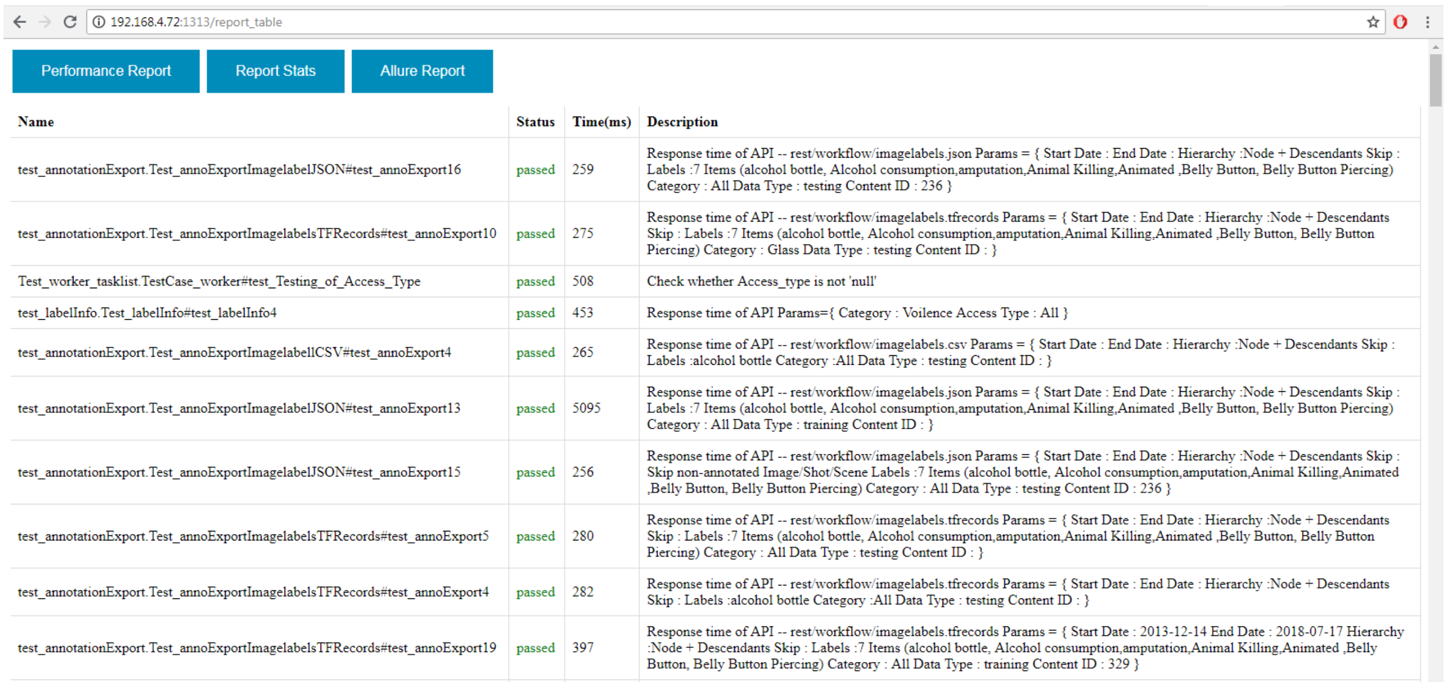
Using .csv file from allure report create **Test stats** and **Test detail** table (Standalone HTML). Update these table daily using python script.

Test Stats



Story	Feature	PASSED	BROKEN	FAILED	SKIPPED	UNKNOWN
Annotation Export (Image label JSON)	Response Time	20	1	0	0	0
Annotation Export (Image label TF Record)	Response Time	21	0	0	0	0
	test case	18	0	3	0	0
Label Info	Response Time	18	0	0	0	0
Annotation Export (Image label CSV)	Response Time	19	2	0	0	0
Annotation Export (GetAllSceneAnnotation)	Response Time	19	2	0	0	0
Annotation Export (SceneLabelsCsv)	Response Time	21	0	0	0	0
Content/Image Export (Content Info)	Response Time	14	0	0	0	0
Annotation Expaort (Image level CSV)	Response Time	1	0	0	0	0
	TOTAL	151	5	3	0	0

Test Details



Name	Status	Time(ms)	Description
test_annotationExport.Test_annoExportImagelabelJSON#test_annoExport16	passed	259	Response time of API -- rest/workflow/imagelabels.json Params = { Start Date : End Date : Hierarchy :Node + Descendants Skip : Labels :7 Items (alcohol bottle, Alcohol consumption,amputation,Animal Killing,Animated ,Belly Button, Belly Button Piercing) Category : All Data Type : testing Content ID : 236 }
test_annotationExport.Test_annoExportImagelabelsTFRecords#test_annoExport10	passed	275	Response time of API -- rest/workflow/imagelabels.tfrecords Params = { Start Date : End Date : Hierarchy :Node + Descendants Skip : Labels :7 Items (alcohol bottle, Alcohol consumption,amputation,Animal Killing,Animated ,Belly Button, Belly Button Piercing) Category : Glass Data Type : testing Content ID : }
Test_worker_tasklist.TestCase_worker#test_Testing_of_Access_Type	passed	508	Check whether Access_type is not 'null'
test_labelInfo.Test_labelInfo#test_labelInfo4	passed	453	Response time of API Params={ Category : Voilence Access Type : All }
test_annotationExport.Test_annoExportImagelabelCSV#test_annoExport4	passed	265	Response time of API -- rest/workflow/imagelabels.csv Params = { Start Date : End Date : Hierarchy :Node + Descendants Skip : Labels :alcohol bottle Category :All Data Type : testing Content ID : }
test_annotationExport.Test_annoExportImagelabelJSON#test_annoExport13	passed	5095	Response time of API -- rest/workflow/imagelabels.json Params = { Start Date : End Date : Hierarchy :Node + Descendants Skip : Labels :7 Items (alcohol bottle, Alcohol consumption,amputation,Animal Killing,Animated ,Belly Button, Belly Button Piercing) Category : All Data Type : training Content ID : }
test_annotationExport.Test_annoExportImagelabelJSON#test_annoExport15	passed	256	Response time of API -- rest/workflow/imagelabels.json Params = { Start Date : End Date : Hierarchy :Node + Descendants Skip : Skip non-annotated Image/Shot/Scene Labels :7 Items (alcohol bottle, Alcohol consumption,amputation,Animal Killing,Animated ,Belly Button, Belly Button Piercing) Category : All Data Type : testing Content ID : 236 }
test_annotationExport.Test_annoExportImagelabelsTFRecords#test_annoExport5	passed	280	Response time of API -- rest/workflow/imagelabels.tfrecords Params = { Start Date : End Date : Hierarchy :Node + Descendants Skip : Labels :7 Items (alcohol bottle, Alcohol consumption,amputation,Animal Killing,Animated ,Belly Button, Belly Button Piercing) Category : All Data Type : testing Content ID : }
test_annotationExport.Test_annoExportImagelabelsTFRecords#test_annoExport4	passed	282	Response time of API -- rest/workflow/imagelabels.tfrecords Params = { Start Date : End Date : Hierarchy :Node + Descendants Skip : Labels :alcohol bottle Category :All Data Type : testing Content ID : }
test_annotationExport.Test_annoExportImagelabelsTFRecords#test_annoExport19	passed	397	Response time of API -- rest/workflow/imagelabels.tfrecords Params = { Start Date : 2013-12-14 End Date : 2018-07-17 Hierarchy :Node + Descendants Skip : Labels :7 Items (alcohol bottle, Alcohol consumption,amputation,Animal Killing,Animated ,Belly Button, Belly Button Piercing) Category : All Data Type : training Content ID : 329 }

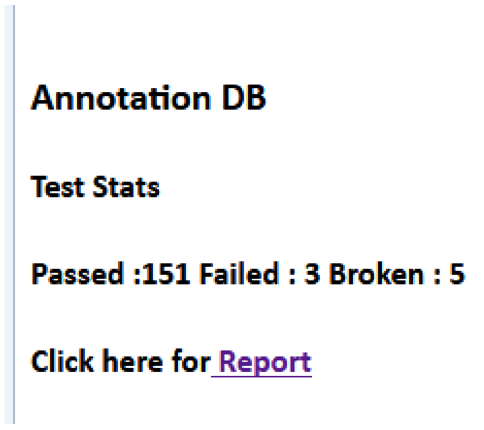
3.14 Task 14

Python Script to email test stats.
Add this Script to Windows task scheduler.

```
from report_stats import reportstats
import smtplib
from email.mime.text import MIMEText
from email.mime.multipart import MIMEMultipart
import json
temp = reportstats()
statdict = {"passed": 0,
            "broken": 0,
            "failed" :0,
            "skip":0,
            "unknown":0
            }

reportstat= json.loads(temp.reportstat())
for item in reportstat:
    statdict["passed"]+= int(item["PASSED"])
    statdict["broken"]+= int(item["BROKEN"])
    statdict["failed"]+= int(item["FAILED"])
    statdict["skip"]+= int(item["SKIPPED"])
    statdict["unknown"]+= int(item["UNKNOWN"])

print(statdict)

msg = MIMEMultipart('alternative')
msg['Subject'] = "Annotation DB Test Report"
part2=("Passed :" + str(statdict["passed"]) + "\t Failed :  " +
       str(statdict["failed"])+  "\t Broken :  " + str(statdict["broken"])  )
html = """
<html>
<head></head>
  <body>
  <h3> Annotation DB </h3>

  <h4>Test Stats </h4>

   <h4> {code}</h4>

    <h4>Click here for<a href="http://192.168.4.72:1313/report_stats">
Report</a></h4>
  </body>
</html>
""".format(code=part2)
```

```
part1=MIMEText(html, 'html')
print(part2)
msg.attach(part1)

s = smtplib.SMTP('interra SMTP Server', 587)
s.starttls()
s.login("you@mail.com”, "*********")
s.sendmail("sender@mail.com", "receiver@mail.com",  msg.as_string())
s.quit()
print("mail successfully sent")
```

Email



Annotation DB

Test Stats

Passed :151 Failed : 3 Broken : 5

Click here for Report

CHAPTER 4

4.1 Conclusion

It was wonderful and learning experience for me while working at Interra Systems.This Project(AnnoDB) took me through various phases of development and gave me real insight into the world of software engineering. The joy of working and the thrill involved while tackling various problems and challenges gave me a feel of developers industry.It was due to this project I came to know how professional software's are designed.

The knowledge that I gained has given me a new perspective which cannot be achieved in a college scenario and helped me realize the realistic importance of the knowledge that we gain in our classrooms.

Overall, I enjoyed each and every bit of work I had put into this Project.

CHAPTER 5

5.1 References

- https://stackoverflow.com
- https://github.com/
- https://www.djangoproject.com/
- https://www.python.org/
- https://www.w3schools.com/
- https://docs.qameta.io/allure/

THANK
YOU
GRACIAS
ARIGATO
SHUKURIA
DANKSCHEEN
TASHAKKUR ATU
YAQHANYELAY
SUKSAMA
EKHMET
GRAZIE
MEHRBANI
KOMAPSUMNIDA
MAAKE
PALDIES
GOZAIMASHITA
EFCHARISTO
JUSPAXAR
BOLZİN
MERCI
BİYAN
SHUKRIA
TINGKI