A service-oriented approach to processing the Sloan Digital Sky Survey

Frank Harris, Milind Patil, Akshay Peshave
Computer Science and Electrical Engineering Department
University of Maryland, Baltimore County
1000 Hilltop Circle
Baltimore MD 21250
Email: fraha1@umbc.edu, milindp1@umbc.edu, peshave1@umbc.edu

***Abstract*—We developed a prototype search solution which provisions an amalgamation of processed SDSS data and Guide Star Catalog(GSC) as a proof of concept. Our efforts are directed towards a two point agenda. Firstly, we publish a processed version of the Sloan Digital Sky Survey (SDSS) dataset with the results of an autonomously generated object catalog with multiple calculated parameters. Secondly, we make the data available and searchable on multiple parameters via our web service, while combining it with existing star catalogues. The service-oriented approach will aid in compartmentalization and efficient services delivery via an open web services stack.**

I. Introduction

The Sloan Digital Sky Survey (SDSS) is a broad astronomical survey via a ground-based 2.5 m telescope that has covered nearly one third of the sky in five spectral channels. Scanning these images to identify bright objects accurately is an active research area. Also, available search solutions for stellar images and objects therein are targeted at scientists and people with substantial knowledge of the domain.

We believe that provisioning a richer search experience would enable expansion of the user base across astronomy expertise levels. Furthermore an effort can be made in the direction of stellar image processing and object identification and classification which would add sheen to the search solution.

The primary inspiration for this project is SkyServer, a web service provided by SDSS that provides access to raw SDSS data as well as visible images and analytic data. The analysis of objects in their catalogue is driven manually by astronomers. We intend to improve on this service in two chief aspects. First, our image processing pipeline is completely autonomous, which we hope will enable a significant increase in the size of the dataset, and as our software improves in speed, may even allow on-the-fly retrieval and processing and remove the need for pre-processing. It is our hope that our research will result in some novel approaches to some of the problems in astronomical image processing. Second, we expand on the existing services by adding the ability to sort and search objects by constellation, and in the future, to use the users location and local time to allow queries on objects based on their visibility to the user. From the image processing perspective, we found that after we had independently developed our technique for identifying bright objects within an image, an existing publication had outlined a very similar approach: “A novel thresholding method for automatically detecting stars in astronomical images”.

We believe that this service will be of interest and utility to both amateur and professional astronomers, as well as educators, statisticians, and computer scientists interested in image processing research.

II. Datasets

A. Guide Star Catalog (GSC)

The GSC is a star catalog compiled to support the Hubble Space Telescope with targeting off-axis stars. We have indexed in excess of 18.5 million stars, binary stars and non-stellar objects from GSC 1.2 with apparent magnitudes ranging from (approximately) 0 to 17. This is the first full sky star catalog created specifically for navigation in outer space.

Objects indexed	18834488
Stars	15361466
Non-stellar Objects	3473022
Cluster of objects	5381931
Magnitude range	0.0 to 17.26

TABLE I: Parameters of GSC release 1.2

B. Sloan Digital Sky Survey (SDSS)

In 2000, this survey began collecting data with a 2.5 m optical telescope at the Apache Point Observatory in New Mexico. It collects images in drifting mode, so that the telescope remains stationary and collects new images as the Earth rotates. Data is recorded on an array of 30 2048x2048 CCDs, in five rows and six columns. Each row is covered with a distinct filter, while the columns provide angular separation.

The data production rate is approximately 200 GB per night of observation. Our research uses the SDSS Data Release 7 (DR7), which in total consists of 15.7 TB of images. The images have an approximate angular resolution of 0.4 arcseconds, and are taken in five spectral channels that range from 298 nm to 1123 nm, covering the near-infrared, visible, and near-ultraviolet range.

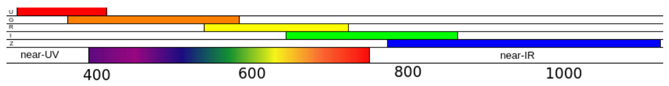


Fig. 1: The approximate range of wavelengths of photons detected for each of the five spectral channels: From top to bottom, u, g, r, i, z.

It is interesting to consider an approximate upper bound on the size of the gridded product that could be produced from DR7. This is a generous upper bound, given that at a latitude of $32°46'49''$ N, There are approximately 9000 pixels to the degree, so a full map of the celestial sphere would be $(9000 \times 360) \times (9000 \times 180) = 5.2488$ terapixels. Each pixel requires 10 bytes, with one 16-bit integer per channel, so that full coverage of the celestial sphere would require $5.2488 \times 10^{12} \times 10 = 52$ terabytes. This appears reasonable, as DR7 is considered to have mapped approximately one third of the night sky.

III. Image Processing and Analysis

For each frame in each column of the run to be processed, the image file in the g spectral channel, as well as the field parameter file, was downloaded from the DAS. The file names follow the naming convention fpC-RRRRRR-gC-FFFF.fit, and tsField-RRRRRR-C-rr-FFFF.fit, where:
RRRRRR: Six digit run number
rr: Non-truncated rerun number
g: The spectral channel, either *g*, *i*, *r*, *u*, or *z*
C: Column number
FFFF: Four digit frame number

In the runs we tested, there were on average about 500 frames, for a total average of 3000 images per run. The first step of processing was to convert the FITS files into intermediate text and binary files, so that the standalone executables and scripts we developed could easily extract metadata and read the arrays. In this way, the code that was dependent on the cfitsio library was minimized, and could mostly run using the C standard library.

After converting the FITS image into a binary array file, an object mask is generated by thresholding. Every pixel that is below the threshold is given a value of 0, while every pixel above the threshold is given a value of 1. Selecting a threshold proved difficult at first. If the value is too high, then legitimate objects will be too low, and will not be recognized. If the value is too low, then in the best case, noisy artifacts or background radiation will be included as objects, and in the worst case, the recursive portion of the object detection algorithm may overflow the stack. Initially, this threshold was chosen experimentally by trying multiple values on multiple images until one was found that allowed a large number of objects on each without letting in excessive background. However, it was discovered that there was too much variation in the brightness distribution across all images, and it did not appear that one threshold would be suitable or optimal for all. A more robust method was necessary to determine the optimal threshold for each image individually and autonomously. We decided to calculate the histogram of the brightness values for each pixel in the image. It was experimentally determined that the brightest two percent of pixels would produce a satisfactory threshold, so our scripts were modified to generate the histogram and calculate the threshold for each image between the format conversion and object detection stages.

Once the mask has been created, this file is modified so that all adjacent sets of pixels share the same unique integer ID. This is done by iterating through the mask and, whenever a pixel with a value of 1 is found, a recursive flood-fill algorithm begins at that position, until all consecutive pixels are assigned a new integer value, which is then incremented for the next object. Since all of the pixels in this object have the value of the identifier assigned to them, and not 1, the object will not be counted again in this algorithm. At the same time of identification, the bounding box of this object is recorded so that the information of interest can be calculated on the subset of the image relevant to the object.

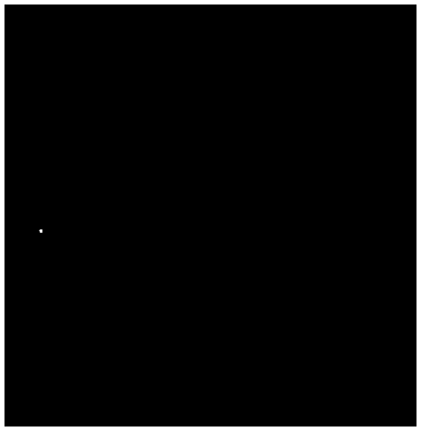
Given the bounding box $(x_{\text{min}}, x_{\text{max}}, y_{\text{min}}, y_{\text{max}})$, the pixel defined as the object center is simply $(\frac{1}{2}(x_{\text{max}} - x_{\text{min}}), \frac{1}{2}(y_{\text{max}} - y_{\text{min}}))$. Calculating right ascension and declension of the object center is a simple addition process, given the index of the object center pixel (x, y) and several values given in the image metadata:

$$\text{ra} = \text{ra}_0 + \frac{\delta}{\delta x}\text{ra}(x - x_0) + \frac{\delta}{\delta y}\text{ra}(y - y_0) \qquad (1)$$

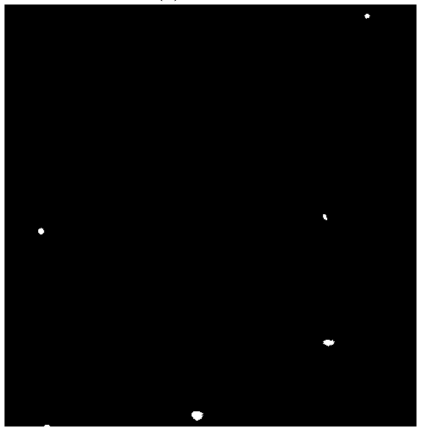
$$\text{dec} = \text{dec}_0 + \frac{\delta}{\delta x}\text{dec}(x - x_0) + \frac{\delta}{\delta y}\text{dec}(y - y_0) \qquad (2)$$

where (x_0, y_0) provide the index of the reference pixel, $(\text{ra}_0, \text{dec}_0)$ give the position of the center of the reference pixel in J2000.0 coordinates, and the delta values demonstrate the change in right ascension and declension for each change in pixel index with respect to x and y. Next, the position for each edge pixel of the object is calculated in the same manner, the angular distance between this point and the center is calculated in degrees, and the minimum and maximum distances are recorded. These minimum and maximum distances can be treated as the magnitudes of the semimajor and semiminor axes of an ellipse. In addition, the position of the center pixel and the most distant pixel can be used to calculate an angle of rotation, where $\theta_{\text{rot}} = 0$ when the major axis is parallel to the coordinate lines of equal declension. Approximating each object as an ellipse can enable us to make some simplifying assumptions which may help in future efforts to perform autonomous object classification.

The solid angle of the object can be calculated in two ways: first by using the assumption of the ellipse. Given the angle of the major and minor axes as θ_1 and θ_2, the solid angle is



(a) T=1500



(b) T=1150

(c) T=1092

Fig. 2: These mask subsets were generated from the same image with a threshold of (i) 1500, (ii) 1150, and (iii) 1092; the last chosen experimentally as the lowest threshold before the algorithm faults. The total images were found to contain 36, 83, and 381 objects, respectively.

$$\Omega = \frac{\pi}{4}\theta_1\theta_2 \tag{3}$$

It can also be done by using the delta values for each pixel, so that

$$d\Omega = \sqrt{(\frac{\delta}{\delta x}\mathrm{ra})^2 + (\frac{\delta}{\delta y}\mathrm{ra})^2} \times \sqrt{(\frac{\delta}{\delta x}\mathrm{dec})^2 + (\frac{\delta}{\delta y}\mathrm{dec})^2} \tag{4}$$

In this way, the solid angle can be approximated by a uniform Riemann sum, multiplying $d\Omega$ by the count of pixels in the object mask. Both solid angle approximations were recorded; in the future we hope to compare, validate, and analyze these values to determine the correctness of both approaches.

After the geometric properties of the object have been calculated, we can now calculate the magnitude. The value we calculated is the Pogson magnitude with respect to the g channel, defined as $M = -2.5\log(\frac{f}{f_0})$; as the project is expanded to the full *ugriz* spectrum, all magnitudes will be calculated separately, and combined to provide a magnitude across the visible spectrum. This magnitude is calculated by summing the values of the pixels that match the integer identifier of the object, subtracting the known bias of 1000, and using the known constant value for exposure time, and given metadata values for zeropoint (z), the extinction coefficient (k), and airmass (a). The flux ratio is given by:

$$\frac{f}{f_0} = \frac{\mathrm{counts}}{\mathrm{exptime}} \times 10^{0.4\times(z+k*a)} \tag{5}$$

After all of these values are calculated for the object, they are written to a comma-separated text file, including the path to the image file. This text file is then read into the SOLR index.

IV. Architecture

Our objectives of this project were primarily to process the SDSS data into a universal format of metadata and to try and link it up with existing catalogs, and then to create a webservice that will simply provide contextual output from the various sources based on the query parameters of an end-user.

To achieve this, we went about designing a highly modular and flexible architecture, in which we can place and replace components very easily, not just to add new features but also to achieve scalability.

The 18.5 million GSC objects are indexed in 4 SOLR cores containing approximately 4.4 million objects each. This multi-core approach is adopted as a load balancing measure to ensure sub-half-second response time from the search service.

The SDSS data (about 100 GB) for our project work was processed using Linux scripts and C code, primarily because our algorithms for image processing kept changing. In the future, we intend to completely automate the processing over the Bluegrit cluster and automatically import that data into our SOLR instances.

The image processing and object identification module provides objects identified per image along with the images metadata extracted from the SDSS FITS files (stored on the bluegrit

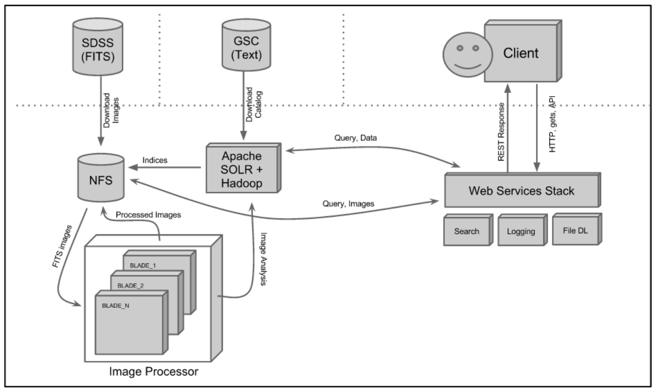


Fig. 3: High-level architecture diagram

NFS). The image metadata is stored in a MySQL database, whereas the identified objects along with their metadata are indexed using a SOLR multicore architecture themselves. This data is critical to our solution as our autonomous search experience provisions a search service on top of this data.

The GSC is merely a data store for comparing our object identification algorithm and provisioning any objects that may not be identified by our algorithm during its stages of development. An important fact is, unlike GSC, our algorithm attempts to identify stellar objects far dimmer than those cataloged in the GSC. Consequently the number of objects is substantially large as compared to the GSC index, thus requiring careful architecting in the long run. We are unable to state exact figures since we have processed only a few runs of the SDSS data as a proof of concept to draw a comparison between the GSC objects and objects identified by our algorithm.

Also we have processed runs from only one of the five channels provided by the SDSS. Once we process all runs we will need to index data from each channel on a separate SOLR multicore framework to meet the sub-second search response objective. The service stack will thus need to undergo subtle alterations to keep up with the new data and architectural changes we may need to effect as the system scales.

The web services stack is hosted on a apache tomcat server and communicates with our SOLR (hosted internally) using SOLR client. It also connects to a mysql server which is used only for storing user authentication data. The web-service is a completely stateless RESTful service and is designed so that we can scale it for performance by simply adding more tomcats and load-balancing the incoming requests.

V. Web Services

In this section, we discuss only the services we have created and exposed. All the services are completely stateless (hence easily load balanceable), authenticated, and operate over both HTTP Get and Post and respond to the user in a variety of formats (JSON, XML currently implemented). Samples of calls and responses are shown in the appendix; we talk only about the API, inputs and output here. The Web Application Description Language (WADL) file is attached in the appendix for clients to automatically integrate into our web service.

Since every API is authenticated, there are two common parameter inputs for each of them:

1) `userid`: Passed as the text `userid` string as is.
2) `password`: Passed as plaintext password string (use HTTPS for more security).

There are some additional optional parameters which are independent of the specific APIs but are handled globally.

3) `format`: takes value as either JSON or XML. If parameter is not passed, then the default response format is JSON.
4) `version`: The default version is 1.0. If we release more versions of the same APIs pass that version, we will try not to remove old APIs completely, but just mark them deprecated while continuing services on it.
5) `start`: default 0. The starting offset in the response of any API, i.e. if API will find 100 values, and start is 50, then the API will respond from offset 50 in the result list to the end of response.
6) `rows`: default infinity. This is used in combination with the start parameter. API will respond [start,start+rows] entries from the results it finds.

Almost all of our APIs query using specific ascension, declension and magnitude values (all double precision), which are:

7) `dblRightAscensionMin`: the minimum ascension value
8) `dblRightAscensionMax`: the maximum ascension value
9) `dblDeclensionMin`: the minimum declension value
10) `dblDeclensionMax`: the maximum declension value
11) `fltMagnitudeMin`: the minimum magnitude value
12) `fltMagnitudeMax`: the maximum magnitude value

A short note about our response. We handle all application exceptions, so our responses are always well formatted and machine decipherable. Every response has 4 key fields:

(a) status: One of SUCCESS or ERROR
(b) code: is 0 if status is success, else one of the errors from the code-error table
(c) cause: the corresponding reason in plaintext for a specific error code
(d) body: this field will be empty if status is error, and will be a well-formatted object if success, containing details of the query output.

The APIs that our service stack currently provisions are:

(a) Search for SDSS Objects in a Given Grid
  - inputs: dblRightAscensionMin, dblRightAscensionMax, dblDeclensionMin, dblDeclensionMax, fltMagnitudeMin, fltMagnitudeMax.
  - response: list of metadata of the SDSS objects for which search is hit.

(b) SDSS Image Search
  - inputs: dblRightAscensionMin, dblRightAscensionMax, dblDeclensionMin, dblDeclensionMax.

- response: list of images from processed SDSS data.

(c) All SDSS objects in an Image
  - inputs: image name whose SDSS objects are to be returned.
  - response: list of all SDSS objects and metadata.

(d) 2 additional APIs to query GSC data
  - these extend the first 2 APIs by including magnitude search and some other parameters for future usage.

VI. User Interface

As a proof of concept of the usage of data that we generate and our data output only architecture, we built an ASP.NET webapp prototype. This prototype utilizes

- SDSS processed and masked image search based on RA and DEC value ranges.
- GSC object search based on the same ranges.
- SDSS identified objects search based on the same ranges.

We also provided a logical overlay grid on an image and the facility to dynamically search for objects (both in GSC and processed SDSS data) by clicking on any cell of the overlay grid. This enables us to gauge the user experience in terms of response times for such a search whose frequency is substantially larger than a plain vanilla image search. To support such use of our APIs the multi-core SOLR architecture is justified.

This prototype also enabled us to understand the overlaps and differences in the object set cataloged in the GSC and those that we identified by processing the SDSS data. This allows us a chance to identify room for improvement in our approach.

VII. Security

The application is designed to be entirely open for anyone to use the services. All the APIs we expose, however, are available on both HTTP and HTTPS and are authenticated, primarily to log usage patterns. The password is stored in our MySQL database as the SHA-1 hash of the password which is salted with a randomly generated string. Everyone should learn from LinkedIns mistake.

```
Stored_Password_Hash =
SHA1(Plain_Text_Password +
Random_Salt_Per_User)
```

Since the only outside network facing component are our API hosting server, and because of our modular approach to the architecture, we propose to move only our tomcat server in the DMZ (De-Militarized Zone) of the network and follow standard hardening practices for it.

VIII. Conclusions and Future Work

This is a preliminary prototype, and has yet to be carefully evaluated for correctness, or given any statistical treatment to summarize the findings. Our work so far suffers from a small data set of only one run, and as such we had difficulty in identifying many particularly bright objects that were likely to be found in another catalog for validation purposes.

Some areas in which we can continue research include:

- Validation of results via comparison to existing catalog
- Expand dataset: all 5 *ugriz* color channels
- Make images more easily viewable with bias subtraction, gamma correction, etc: We made the FITS files viewable using a homebrewed program which converted the arrays into `.ppm` files, and then used the UNIX `convert` command to convert them into `.png` files. Adding color functionality by semi-arbitrarily assigning three-channel weights to each of the five channels, and adding gamma capability to this code, will make the images, and not just the masks, more easily viewable.
- Convert 1-channel single images into gridded, stitched 3D images.
- Statistical analysis of dataset as a whole.
- Continue to develop new ID techniques; analyze object shapes and utilize color information
- Improve performance time via parallel computing
- More interesting queries: Matching GSC/SDSS objects may prove to be a place of challenging research, especially when exact positional information does not agree.

References

[1] M. J.E., R. S., M. B., B. B., and L. B., "The guide star catalog version 1.2: an astrometric recalibration and other refinements." *Astronomical Journal*, vol. 121, 2001. [Online]. Available: http://cdsarc.u-strasbg.fr/viz-bin/Cat?bincats/GSC_1.2

[2] M. Albrecht, *Encoding the GSC*, 1996. [Online]. Available: http://archive.eso.org/skycat/gsc-engine/node2.html

[3] H. M., "Web application description language," *W3C Member Submission*, Aug. 2009. [Online]. Available: http://www.w3.org/Submission/2009/SUBM-wadl-20090831/

[4] G. M. *et al.* (2011, Nov.) Jersey and wadl. [Online]. Available: https://wikis.oracle.com/display/Jersey/WADL

[5] Apache solr wiki. [Online]. Available: http://wiki.apache.org/solr/

[6] *.NET Framework 4.5*. [Online]. Available: http://msdn.microsoft.com/en-us/library/w0x726c2(v=VS.110).aspx

[7] S. A., G. J., T. A., B. B., G. R., L. N., K. P., M. T., O. W., N.-S. M., R. J., S. C., and vandenBerg J., "The sdss dr1 skyserver: Public access to a terabyte of astronomical data," soon to be published in a technical journal at unknown date. [Online]. Available: http://skyserver.sdss.org/public/en/skyserver/paper/

[8] e. a. Abazajian, "The seventh data release of the sloan digital sky survey," *Astrophysical Journal Supplement Series*, vol. 182, no. 543, 2009. [Online]. Available: http://adsabs.harvard.edu/abs/2009ApJS..182..543A

[9] Sdss data release 7. [Online]. Available: http://www.sdss.org/dr7/

[10] V. D. Cristo A., Plaza A., "A novel thresholding method for automatically detecting stars in astronomical images," in *IEEE International Symposium on Signal Processing and Information Technology*, 2008. [Online]. Available: http://www.researchgate.net/publication/224381182_A_novel_thresholding_method_for_automatically_detecting_stars_in_astronomical_images

[1] [2] [3] [4] [5] [6] [7] [8] [9] [10]

APPENDIX A
WEB APPLICATION DESCRIPTION LANGUAGE (WADL) FOR THE SERVICE

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<application xmlns="http://research.sun.com/wadl/2006/10">
  <doc xmlns:jersey="http://jersey.dev.java.net/" jersey:generatedBy="Jersey: 1.3
      06/17/2010 04:53 PM"/>
  <resources base="http://127.0.0.1:8080/SDSS/">
    <resource path="/rest">

      <resource path="method1">
        <method id="service" name="GET">
          <request>
            <param xmlns:xs="http://www.w3.org/2001/XMLSchema" name="userid" style="
                query" type="xs:string" required="true"/>
            <param xmlns:xs="http://www.w3.org/2001/XMLSchema" name="password" style
                ="query" type="xs:string" required="true"/>
            <param xmlns:xs="http://www.w3.org/2001/XMLSchema" name="format" style="
                query" type="xs:string" required="false"                    default
                ="json"/>
            <param xmlns:xs="http://www.w3.org/2001/XMLSchema" name="start" style="
                query" type="xs:int" required="false" default="0"/>
            <param xmlns:xs="http://www.w3.org/2001/XMLSchema" name="rows" style="
                query" type="xs:int" required="false" default="100"/>
            <param xmlns:xs="http://www.w3.org/2001/XMLSchema" name="
                dblRightAscensionMin" style="query" type="xs:double" required="true"/>
            <param xmlns:xs="http://www.w3.org/2001/XMLSchema" name="
                dblRightAscensionMax" style="query" type="xs:double" required="true"/>
            <param xmlns:xs="http://www.w3.org/2001/XMLSchema" name="dblDeclensionMin
                " style="query" type="xs:double" required="true"/>
            <param xmlns:xs="http://www.w3.org/2001/XMLSchema" name="dblDeclensionMax
                " style="query" type="xs:double" required="true"/>
            <param xmlns:xs="http://www.w3.org/2001/XMLSchema" name="fltMagnitudeMin"
                 style="query" type="xs:double" required="true"/>
            <param xmlns:xs="http://www.w3.org/2001/XMLSchema" name="fltMagnitudeMax"
                 style="query" type="xs:double" required="true"/>
            <param xmlns:xs="http://www.w3.org/2001/XMLSchema" name="version" style="
                query" type="xs:string" default="1.0" required="false"/>
          </request>
          <response>
            <representation mediaType="application/xml"/>
            <representation mediaType="application/json"/>
          </response>
        </method>
      </resource>

      <resource path="method2">
        <method id="service" name="GET">
          <request>
            <param xmlns:xs="http://www.w3.org/2001/XMLSchema" name="userid" style="
                query" type="xs:string" required="true"/>
            <param xmlns:xs="http://www.w3.org/2001/XMLSchema" name="password" style
                ="query" type="xs:string" required="true"/>
            <param xmlns:xs="http://www.w3.org/2001/XMLSchema" name="format" style="
                query" type="xs:string" required="false"                    default
                ="json"/>
            <param xmlns:xs="http://www.w3.org/2001/XMLSchema" name="start" style="
                query" type="xs:int" required="false" default="0"/>
            <param xmlns:xs="http://www.w3.org/2001/XMLSchema" name="rows" style="
                query" type="xs:int" required="false" default="100"/>
            <param xmlns:xs="http://www.w3.org/2001/XMLSchema" name="
                dblRightAscensionMin" style="query" type="xs:double" required="true"/>
            <param xmlns:xs="http://www.w3.org/2001/XMLSchema" name="
                dblRightAscensionMax" style="query" type="xs:double" required="true"/>
            <param xmlns:xs="http://www.w3.org/2001/XMLSchema" name="dblDeclensionMin
                " style="query" type="xs:double" required="true"/>
            <param xmlns:xs="http://www.w3.org/2001/XMLSchema" name="dblDeclensionMax
                " style="query" type="xs:double" required="true"/>
            <param xmlns:xs="http://www.w3.org/2001/XMLSchema" name="version" style="
                query" type="xs:string" default="1.0" required="false"/>
          </request>
          <response>
            <representation mediaType="application/xml"/>
            <representation mediaType="application/json"/>
          </response>
        </method>
      </resource>

      <resource path="method3">
ation mediaType="application/xml"/>
            <representation mediaType="application/json"/>
          </response>
        </method>
      </resource>

    </resource>

  </resources>
</application>        <method id="service" name="GET">
          <request>
            <param xmlns:xs="http://www.w3.org/2001/XMLSchema" name="userid" style="
                query" type="xs:string" required="true"/>
            <param xmlns:xs="http://www.w3.org/2001/XMLSchema" name="password" style
                ="query" type="xs:string" required="true"/>
            <param xmlns:xs="http://www.w3.org/2001/XMLSchema" name="format" style="
                query" type="xs:string" required="false"                    default
                ="json"/>
            <param xmlns:xs="http://www.w3.org/2001/XMLSchema" name="start" style="
                query" type="xs:int" required="false" default="0"/>
            <param xmlns:xs="http://www.w3.org/2001/XMLSchema" name="rows" style="
                query" type="xs:int" required="false" default="100"/>
            <param xmlns:xs="http://www.w3.org/2001/XMLSchema" name="
                dblRightAscensionMin" style="query" type="xs:double" required="true"/>
            <param xmlns:xs="http://www.w3.org/2001/XMLSchema" name="
                dblRightAscensionMax" style="query" type="xs:double" required="true"/>
            <param xmlns:xs="http://www.w3.org/2001/XMLSchema" name="dblDeclensionMin
                " style="query" type="xs:double" required="true"/>
            <param xmlns:xs="http://www.w3.org/2001/XMLSchema" name="dblDeclensionMax
                " style="query" type="xs:double" required="true"/>
            <param xmlns:xs="http://www.w3.org/2001/XMLSchema" name="version" style="
                query" type="xs:string" default="1.0" required="false"/>
          </request>
          <response>
            <representation mediaType="application/xml"/>
            <representation mediaType="application/json"/>
          </response>
        </method>
      </resource>

      <resource path="method4">
        <method id="service" name="GET">
          <request>
            <param xmlns:xs="http://www.w3.org/2001/XMLSchema" name="userid" style="
                query" type="xs:string" required="true"/>
            <param xmlns:xs="http://www.w3.org/2001/XMLSchema" name="password" style
                ="query" type="xs:string" required="true"/>
            <param xmlns:xs="http://www.w3.org/2001/XMLSchema" name="format" style="
                query" type="xs:string" required="false"                    default
                ="json"/>
            <param xmlns:xs="http://www.w3.org/2001/XMLSchema" name="start" style="
                query" type="xs:int" required="false" default="0"/>
            <param xmlns:xs="http://www.w3.org/2001/XMLSchema" name="rows" style="
                query" type="xs:int" required="false" default="100"/>
            <param xmlns:xs="http://www.w3.org/2001/XMLSchema" name="
                dblRightAscensionMin" style="query" type="xs:double" required="true"/>
            <param xmlns:xs="http://www.w3.org/2001/XMLSchema" name="
                dblRightAscensionMax" style="query" type="xs:double" required="true"/>
            <param xmlns:xs="http://www.w3.org/2001/XMLSchema" name="dblDeclensionMin
                " style="query" type="xs:double" required="true"/>
            <param xmlns:xs="http://www.w3.org/2001/XMLSchema" name="dblDeclensionMax
                " style="query" type="xs:double" required="true"/>
            <param xmlns:xs="http://www.w3.org/2001/XMLSchema" name="version" style="
                query" type="xs:string" default="1.0" required="false"/>
          </request>
          <response>
            <representation mediaType="application/xml"/>
            <representation mediaType="application/json"/>
          </response>
        </method>
      </resource>

      <resource path="method5">
        <method id="service" name="GET">
          <request>
            <param xmlns:xs="http://www.w3.org/2001/XMLSchema" name="userid" style="
                query" type="xs:string" required="true"/>
            <param xmlns:xs="http://www.w3.org/2001/XMLSchema" name="password" style
                ="query" type="xs:string" required="true"/>
            <param xmlns:xs="http://www.w3.org/2001/XMLSchema" name="format" style="
                query" type="xs:string" required="false"                    default
                ="json"/>
            <param xmlns:xs="http://www.w3.org/2001/XMLSchema" name="version" style="
                query" type="xs:string" default="1.0" required="false"/>
            <param xmlns:xs="http://www.w3.org/2001/XMLSchema" name="imageName" style
                ="query" type="xs:string" required="true"/>
          </request>
          <response>
            <represent
```

APPENDIX B
WEB SERVICE DESCRIPTION LANGUAGE (WSDL) SPECIFICATION FOR THE SERVICE

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!-- definition is replaced by description in WSDL 2.0 -->
<wsdl:description xmlns:wsdl="http://www.w3.org/ns/wsdl"
                  targetNamespace="http://127.0.0.1:8081/SDSS/wsdl"
   xmlns:tns="http://127.0.0.1:8081/SDSS/wsdl"
   xmlns:whttp="http://www.w3.org/ns/wsdl/http"
   xmlns:wsdlx="http://www.w3.org/ns/wsdl-extensions"
   xmlns:xs="http://www.w3.org/2001/XMLSchema">

    <wsdl:documentation>
      This is a WSDL 2.0 description of the SDSS project for SOC 668 class.
    </wsdl:documentation>

<wsdl:types>
    <xs:import namespace="http://127.0.0.1:8081/SDSS/xsd"
        schemaLocation="SDSSMethod1.xsd"/>
</wsdl:types>

  <!-- portType is replaced by interface , also message is now encapsulated in
      interface -->
  <wsdl:interface name="Method1Interface">
    <wsdl:operation name="method1"
        pattern="http://www.w3.org/ns/wsdl/in-out"
        style="http://www.w3.org/ns/wsdl/style/iri"
        wsdlx:safe="true">
      <wsdl:documentation>
          This operation returns a list of SDSS objects.
      </wsdl:documentation>
      <wsdl:input element="msg:getParameters"/>
      <wsdl:output element="msg:response"/>
    </wsdl:operation>
  </wsdl:interface>

  <wsdl:binding name="Method1HTTPBinding"
      type="http://www.w3.org/ns/wsdl/http"
      interface="Method1Interface">
    <wsdl:documentation>
        The RESTful HTTP binding for the SDSS service.
    </wsdl:documentation>
    <wsdl:operation ref="tns:getParameters" whttp:method="GET"/>
  </wsdl:binding>
  <!-- binding is now reusable -->


  <wsdl:service name="METHOD1" interface="Method1Interface">
    <wsdl:documentation>
        The SDSS's main service
    </wsdl:documentation>
    <wsdl:endpoint name="METHOD1"
        binding="tns:Method1HTTPBinding"
        address="http://127.0.0.1:8081/SDSS/method1/">
    </wsdl:endpoint>
  </wsdl:service>
```

```
</wsdl:description>
```

A. Corresponding XML Schema Document (XSD)

```
<schema xmlns="http://www.w3.org/2001/XMLSchema"
    targetNamespace="http://127.0.0.1:8081/SDSS/xsd"
    xmlns:tns="http://127.0.0.1:8081/SDSS/xsd"
    xmlns:sdssvc="http://127.0.0.1:8081/SDSS/wsdl"
    xmlns:wsdlx="http://www.w3.org/ns/wsdl-extensions"
    elementFormDefault="qualified">

  <element name="getParameters" type="tns:getParametersType">
    <annotation>
        <documentation>
         The request element for the SDSS service.
        </documentation>
     </annotation>
  </element>

  <element name="response" type="tns:responseType">
    <annotation>
      <documentation>
         The response element for the SDSS service.
      </documentation>
    </annotation>
  </element>

  <complexType name="getParametersType">
    <sequence>
      <element name="userid" type="string"/>
      <element name="password" type="string"/>
      <element name="format" type="string"/>
      <element name="start" type="integer"/>
      <element name="rows" type="integer"/>
      <element name="dblRightAscensionMin" type="double"/>
      <element name="dblRightAscensionMax" type="double"/>
      <element name="dblDeclensionMin" type="double"/>
      <element name="dblDeclensionMax" type="double"/>
      <element name="fltMagnitudeMin" type="double"/>
      <element name="fltMagnitudeMax" type="double"/>
      <element name="version" type="string"/>
    </sequence>
  </complexType>

  <complexType name="responseType">
    <sequence>
      <element name="myResponse" type="tns:myResponseType" minOccurs="1" maxOccurs
          ="1"/>
    </sequence>
  </complexType>

  <complexType name="myResponseType">
    <attribute name="body" type="string" minOccurs="0" maxOccurs="1"/>
    <attribute name="status" type="string" />
    <attribute name="cause" type="string" />
    <attribute name="code" type="integer" />
  </complexType>

</schema>
```

Appendix C
Sample Usage and Output

1) With incorrect/missing userid/password:

```
http://127.0.0.1:8081/SDSS/rest/method1?userid=helloandpassword=password

{"body":"","cause":"userid or password missing or mismatch","status":"ERROR","code
    ":"101"}

http://127.0.0.1:8081/SDSS/rest/method1?userid=helloandpassword=passwordandformat=
    xml

<myResponse><body/><cause>userid or password missing or mismatch</cause><status>
    ERROR</status><code>101</code></myResponse>
```

2) With correct authentication but missing mandatory parameters:

```
http://127.0.0.1:8081/SDSS/rest/method1?userid=milindp1andpassword=password

{"body":"","cause":"missing parameter or empty, please check","status":"ERROR","
    code":"103"}

http://127.0.0.1:8081/SDSS/rest/method1?userid=milindp1andpassword=
    passwordandformat=xml

<myResponse><body/><cause>missing parameter or empty, please check</cause><status>
    ERROR</status><code>103</code></myResponse>
```

3) Sample response with valid parameters:

```
http://127.0.0.1:8081/SDSS/rest/method3?anddblRightAscensionMin=0
    anddblRightAscensionMax=200anddblDeclensionMin=-1anddblDeclensionMax=1andstart
    =0androws=10

{
    "body":{
        "response":{
            "start":0,
            "docs":[
                {
                    "longGscID":27201051,
                    "fltMagnitudeError":0.05,
                    "intBand":4,
                    "fltMagnitude":0,
                    "dblRightAscension":176.93329,
                    "dblDeclension":0.80795,
                    "intClass":0,
                    "fltPosError":0.1,
                    "intMultipleObjects":0,
                    "strPlateNum":"+017"
                },
                {
                    "longGscID":469901219,
                    "fltMagnitudeError":0.05,
    "intBand":4,
    "fltMagnitude":0,
    "dblRightAscension":40.69171,
    "dblDeclension":−0.02317,
    "intClass":0,
    "fltPosError":0.1,
    "intMultipleObjects":0,
    "strPlateNum":"+017"
},
{
    "longGscID":481603054,
    "fltMagnitudeError":0.05,
    "intBand":4,
    "fltMagnitude":0,
    "dblRightAscension":108.83088,
    "dblDeclension":−0.16131,
    "intClass":0,
    "fltPosError":0.1,
    "intMultipleObjects":0,
    "strPlateNum":"+017"
},
{
    "longGscID":481503818,
    "fltMagnitudeError":0.05,
    "intBand":4,
    "fltMagnitude":0,
    "dblRightAscension":107.34383,
    "dblDeclension":−0.80656,
    "intClass":0,
    "fltPosError":0.1,
    "intMultipleObjects":0,
    "strPlateNum":"+017"
},
{
    "longGscID":480104464,
    "fltMagnitudeError":0.05,
    "intBand":4,
    "fltMagnitude":0,
    "dblRightAscension":104.65967,
    "dblDeclension":−0.47881,
    "intClass":0,
    "fltPosError":0.1,
    "intMultipleObjects":0,
    "strPlateNum":"+017"
},
{
    "longGscID":476602445,
    "fltMagnitudeError":0.1,
    "intBand":4,
    "fltMagnitude":2.2,
    "dblRightAscension":83.0015,
    "dblDeclension":−0.29908,
    "intClass":0,
    "fltPosError":0.1,
    "intMultipleObjects":0,
    "strPlateNum":"+056"
```

```
        },
        {
            "longGscID":494001158,
            "fltMagnitudeError":0.07,
            "intBand":4,
            "fltMagnitude":3.86,
            "dblRightAscension":184.97646,
            "dblDeclension":−0.66681,
            "intClass":0,
            "fltPosError":0,
            "intMultipleObjects":0,
            "strPlateNum":"+056"
        },
        {
            "longGscID":4601633,
            "fltMagnitudeError":0.05,
            "intBand":4,
            "fltMagnitude":4.05,
            "dblRightAscension":39.87062,
            "dblDeclension":0.3285,
            "intClass":0,
            "fltPosError":0,
            "intMultipleObjects":0,
            "strPlateNum":"+056"
        },
        {
            "longGscID":481503816,
            "fltMagnitudeError":0,
            "intBand":4,
            "fltMagnitude":4.19,
            "dblRightAscension":107.96575,
            "dblDeclension":−0.49281,
            "intClass":0,
            "fltPosError":0.1,
            "intMultipleObjects":0,
            "strPlateNum":"+056"
        },
        {
            "longGscID":493001093,
            "fltMagnitudeError":0,
            "intBand":4,
            "fltMagnitude":4.26,
            "dblRightAscension":174.23721,
            "dblDeclension":−0.82381,
            "intClass":0,
            "fltPosError":0,
            "intMultipleObjects":0,
            "strPlateNum":"+056"
        }
    ],
    "numFound":148245
},
"responseHeader":{
    "status":0,
    "QTime":1,
    "params":{
            "sort":"fltMagnitude asc",
            "start":"0",
            "q":"dblRightAscension:[0 TO 200] AND dblDeclension:[-1 TO 1]",
            "wt":"json",
            "rows":"10"
         }
      }
   },
   "cause":"success",
   "status":"SUCCESS",
   "code":"0"
}
```

```
http://127.0.0.1:8081/SDSS/rest/method2?dblRightAscensionMin=170.141678
    anddblRightAscensionMax=170.2anddblDeclensionMin=-0.99anddblDeclensionMax=-0.91
    andstart=0androws=10
```

```
{
   "body":{
      "value0":{
         "constellation":"psc",
         "theta":-0.010306,
         "count":65280,
         "minorAxis":0.00158,
         "numPixels":127,
         "magnitude":15.799879,
         "rightAscension":170.141678,
         "majorAxis":9.88E-4,
         "solidAngle":0,
         "ojbId":112,
         "declension":-0.913468,
         "eccentricity":0.780163
      },
      "value1":{
         "constellation":"psc",
         "theta":-0.005091,
         "count":74325,
         "minorAxis":0.002294,
         "numPixels":173,
         "magnitude":15.658993,
         "rightAscension":170.156845,
         "majorAxis":6.71E-4,
         "solidAngle":0,
         "ojbId":134,
         "declension":-0.919076,
         "eccentricity":0.95622
      },
      "value2":{
         "constellation":"psc",
         "theta":-0.015296,
         "count":90713,
         "minorAxis":0.001717,
         "numPixels":155,
         "magnitude":15.442656,
         "rightAscension":170.173538,
            ”majorAxis”:0.00108,
            ”solidAngle”:0,
            ”ojbId”:155,
            ”declension”:−0.980973,
            ”eccentricity”:0.777578
        },
        ”size”:3
    },
    ”cause”:”success”,
    ”status”:”SUCCESS”,
    ”code”:”0”
}
```