A PROJECT REPORT ON

ARCHITECTURAL OPTIMIZATION OF ASTROSAT-CZTI DATA QUALITY REPORTS

SUBMITTED BY

Brijesh Choudhary (B150024225)
Ankit Dani (B150024226)
Aditya Gutte (B150024246)
Chinmay Pophale (B150024313)

UNDER THE GUIDANCE OF
Prof. V. S. Jagtap

SPONSORED BY
Inter University Centre for Astronomy and Astrophysics, Pune.

Department Of Computer Engineering
MAEERs MAHARASHTRA INSTITUTE OF TECHNOLOGY
Kothrud, Pune 411 038

2018-2019

Department Of Computer Engineering
MAEERs MAHARASHTRA INSTITUTE OF TECHNOLOGY PUNE

C E R T I F I C A T E

This is to certify that

Brijesh Choudhary (B150024225)

Ankit Dani (B150024226)

Aditya Gutte (B150024246)

Chinmay Pophale (B150024313)

of B. E. Computer successfully completed project in

ARCHITECTURAL OPTIMIZATION OF ASTROSAT-CZTI DATA QUALITY REPORTS

to my satisfaction and submitted the same during the academic year 2018-2019 towards the partial fulfillment of degree of Bachelor of Engineering in Computer Engineering of Pune University under the Department of Computer Engineering , Maharashtra Institute of Technology, Pune.

Prof. V. S. Jagtap
(Project Guide)

Dr. V. Y. Kulkarni
(Head of Computer Engineering Department)

Place: Pune

Date:

ACKNOWLEDGEMENT

We take this opportunity to express our sincere appreciation for the cooperation given by Dr. V. Y. Kulkarni, Head Of Department (Department of Computer Engineering) and need a special mention for all the motivation and support.

We are deeply indebted to our guide Prof. V. S. Jagtap for completion of our project for which she has guided and helped us going out of the way.

For all efforts behind the Project report, We would also like to express our sincere appreciation to staff of department of Computer Engineering, Maharashtra Institute of Technology Pune, for their extended help and suggestions at every stage.

Brijesh Choudhary
(Exam Seat No: B150024225)

Ankit Dani
(Exam Seat No: B150024226)

Aditya Gutte
(Exam Seat No: B150024246)

Chinmay Pophale
(Exam Seat No: B150024313)

Contents

List of Figures

List of Tables

List of Abbreviations

ISRO	Indian Space Research Organisation
CZTI	Cadmium Zinc Telluride Imager
DQR	Data Quality Report
IUCAA	Inter-University Centre for Astronomy and Astrophysics
FITS	Flexible Image Transport System
HTTP	Hyper Text Transfer Protocol
REST	REpresentational State Transfer
SOAP	Simple Object Access Protocol
JSON	JavaScript Object Notation
SQL	Structured Query Language
RDBMS	Relational Database Management System
ASCII	American Standard Code for Information Interchange
JAXP	Java API for XML Processing
SAX	Simple API for XML
DTD	Document Type Definition
ARM	Advanced RISC Machine

SSL	Secure Socket Layer
ART	Android RunTime
CSMA/CA	Carrier Sense Multiple Access/Collision Avoidance
MIMO	Multiple input, Multiple output
IDE	Integrated Development Environment
ARJUN	Access and Retrieval Jointly Using N-tier architecture
URL	Universal Resource Locator
MVT	Model View Template
DFD	Data Flow Diagram
EDI	Electronic Data Exchange

Abstract

Satellites all around the globe generate huge quantities of data periodically on a daily basis. The management of such large quantities of data poses significant problems ranging from data storage, retrieval and manipulation. The current scenario of managing Indian Space Research Organisation's (ISRO) ASTROSAT-CZTI satellite data involves an elementary and unsophisticated approach which causes a lot of delay in data retrieval and further causes noticeable data portability issues. The evaluation and research on the above mentioned complication paved a way for us to introduce a new architectural system to solve this complex optimization problem. We propose the creation of an independent application server and an underlying Django REST API based web service which would facilitate the efficient management and retrieval of this data. This also extends the scope, making the system more dynamic and helps in load balancing. The application server helps the system to regulate the turbulent flow of huge data quantities. The Django web framework provides feature distribution which helps manage unorganised data cost effectively. Thus, it removes the possible causes of deterioration of response in the existing system. The REST API extends the scope of the system by providing umpteen useful features which makes universal interfacing of our system possible. The techniques used in the development of our proposed system including data migration, prefetching of data and minimizing the time trade-off, address all existing challenges and show a significant improvement in the overall performance of the system.

Keywords:
Django, REST, Volley, JSON, MySQL, Migration, Data Management, Web Service, Data Fetching, Astronomy.

Chapter 1

INTRODUCTION

ASTROSAT is India's first dedicated multi-wavelength space observatory. It was launched on a PSLV-XL on 28 September 2015. With the success of this satellite ISRO has proposed to launch AstroSat-2 as a successor for ASTROSAT when nears its five-year life span. A number of astronomy research institutions in India, and abroad have jointly built instruments for the satellite. Important areas requiring coverage include studies of astrophysical objects ranging from nearby solar system objects to distant stars and objects at cosmological distances; timing studies of variables ranging from pulsations of hot white dwarfs to those of active galactic nuclei can be conducted with ASTROSAT as well, with time scales ranging from milliseconds to days. ASTROSAT is a multi-wavelength astronomy mission on an IRS-class satellite into a near-Earth, equatorial orbit. The five instruments on board cover the visible (320530 nm), near UV (180300 nm), far UV (130180 nm), soft X-ray (0.38 keV and 210 keV) and hard X-ray (380 keV and 10150 keV) regions of the electromagnetic spectrum. ASTROSAT was successfully launched on 28 September 2015 from the Satish Dhawan Space Centre on board a PSLV-XL vehicle at 10:00AM.

The Cadmium Zinc Telluride Imager (CZTI) is a hard X-ray imager. It will consist of a pixilated Cadmium-Zinc-Telluride detector array of 500 cm2 effective area and the energy range from 10 to 150 kev. The detectors have a detection efficiency

close to 100% up to 100 keV, and have a superior energy resolution (2% at 60 keV) compared to scintillation and proportional counters. Their small pixel size also facilitates medium resolution imaging in hard x-rays. The CZTI will be fitted with a two dimensional coded mask, for imaging purposes. The sky brightness distribution will be obtained by applying a deconvolution procedure to the shadow pattern of the coded mask recorded by the detector. Apart from spectroscopic studies, CZTI would be able to do sensitive polarization measurements for bright galactic X-ray sources in 100300 keV. ISRO has set up a support cell for ASTROSAT at IUCAA, Pune. A MoU was signed between ISRO and IUCAA in May 2016. The support cell has been set up to give opportunity to the scientific community in making proposals on processing and usage of ASTROSAT data. The support cell will provide necessary resource materials, tools, training and help to the guest observers.

The new architecture for the storage of ASTROSAT DQRs suggests the use of a structured database for storing and managing the data through a web service and availing the data to the end user in an android application. This architecture proposes improved and ubiquitous data access and reduced latency. Optimizing the storage solution of CZTI data by implementing web services, database servers and front end android interface. The web service will extract data from the existing servers via data migration streams and store the data in a specific format that will optimize the retrieval and access of the satellite data and reduce the latency.

1.1 Motivation

The Inter-University Centre for Astronomy and Astrophysics (IUCAA) is an autonomous institution set up by the University Grants Commission to promote nucleation and growth of active groups in astronomy and astrophysics in Indian universities. IUCAA is located in the University of Pune campus. Scientists at IUCAA

carry out research in a wide range of areas in astronomy, astrophysics and physics. IUCAA has active research groups in fields like classical and quantum gravity, cosmology, gravitational waves, optical and radio astronomy, solar system physics and instrumentation.ISRO has set up a support cell for AstroSat at IUCAA, Pune. A MoU was signed between ISRO and IUCAA in May 2016. The support cell has been set up to give opportunity to the scientific community in making proposals on processing and usage of AstroSat data. The support cell will provide necessary resource materials, tools, training and help to the guest observers. To work with such an esteemed organization was the primary motivation for the project. This project was an inter-disciplinary project requiring the use of latest technologies in Computer advancements.

1.2 Problem Definition

ASTROSAT (CZTI-Cadmium Zinc Telluride Imager) data is acquired from the satellite in the FITS(Flexible Image Transmission System) format and stored after transforming it into ASCII and JPEG at IUCAA servers. The current storage of the DQR (Data Quality Reports) is in HTML (Hypertext Markup Language) format on the IUCAA web servers: it generates dynamic web pages for every new orbit added to the database. It takes infeasible time (approximately two minutes on a stable data connection) to load the webpage entirely and then perform the operations. The interface is quite rudimentary and does not support many features. Currently the data is not suitable to be accessed on mobile devices as mobile processing power is not sufficient to process such huge amounts of data. The ASTROSAT generates 15 rows of data each day, each row contains an orbits data of size approximately 85 MB which further aggravates the problem of efficient data storage and access.

1.3 Objective

- The main objective of the project was to design an efficient architecture to get optimised results for fetching ASTROSAT CZTI DQRs
- Another objective is to design a full fledged automated system that will manage the storage and access of the records.
- To provide simplistic and intuitive user interface that will allow the users to understand and comprehend the data with ease.
- The system is designed to have real time data updation as and when the satellite generates the data.
- To provide access to the Astrosat DQRs anywhere at any time.
- To provide high scalability and portability to the astronomical data gathered by Astrosat.

Table 1.2: Comparision of Exisiting and Proposed System

Attributes	**Existing System**	**Proposed System**
Web Services	Absent	Present
Application Server	Not in use	Used
Caching	Absent	Present
Compatibility	Browser only	Android app and browser
Delay Time	High	Low

Chapter 2

LITERATURE SURVEY

Sr no	Paper Title, Authors, year of publication	Concepts described in the paper	Findings from the paper
1	ASTROSAT Handbook Ver 1.9, 2016, Dipankar Bhattacharya, Gulab C Dewangan, IUCAA	General overview of ASTROSAT architecture, its functionalities.	High response time and data retrieval rate.
2	Toward a Reliable Service -Based Approach to Software Application Development, 2018 IEEE 20th Conference on Business Informatics (CBI), Qusay Mahmoud, Ian Andrusiak, May Altaei.	Web Services, Software Development, REST.	Low accessibility rate and high internet bandwidth.
3	Research and implementation of Web Services in Android network communication framework-Volley, IEEE 2014 11th International Conference on Service Systems and Service Management Yang Shulin, Hu Jieping.	Web Services, Protocols, Android Network Communication, Google.	Multi-part Requests

Sr no	**Paper Title, Authors, year of publication**	**Concepts described in the paper**	**Findings from the paper**
4	An implementation of Web Services on mobile terminal, IEEE 2013 6th International Conference on Information Management, Innovation Management and Industrial Engineering, Huiling Hao, Chunlai Zhao, Tianlai Wang.	Simple Object Access Protocol, Mobile Communication.	Too much reliance on HTTP, Slow.
5	Design web service academic information system based multi-platform, IEEE International Conference on Communication, Networks and Satellite, Meta Lara Pandini, Zainal Arifin, Dyna Marisa Khairina	REST, JSON.	No error handling and not fail-safe.
6	A qualitative analysis of the performance of MongoDB vs MySQL database based on insertion and retriewal operations using a web application to explore load balancing Sharding in MongoDB and its advantages, IEEE 2014, International Conference on I-SMAC, M. M. Patil, A. Hanni, C. H. Tejeshwar and P. Patil.	Ascertains the need for NoSQL databases and emphasizes advancement of document -oriented database - MongoDB	Sharding is efficient only on multi-server and distributed systems

Sr no	**Paper Title, Authors, year of publication**	**Concepts described in the paper**	**Findings from the paper**
7	A comparative study: MongoDB vs. MySQL, IEEE 2015 13th International Conference on Engineering of Modern Electric Systems, C. Gyrdi, R.Gyrdi, G. Pecherle.	Presents a comparative study of non-relational databases and relational databases	MongoDB is more efficient yet Data Analysis is easier with MySQL
8	REST Services with Django In: Beginning Django, 2017, Rubio D	Discovers the Django web application framework and getting started building Python-based web applications	Using Django framework for building scalable and efficient web applications
9	Design Patterns and Extensibility of REST API for Networking Applications, IEEE 2016 Transactions on Network and Service Management, L. Li, W. Chou, W. Zhou and M. Luo	This paper addresses the issues in hypertext-driven navigation in REST APIs. Describes some important design patterns, such as backtracking and generator	The proposed cache mechanism reduces the overhead of using the hypertext -driven REST API by 66 percent.

Sr no	Paper Title, Authors, year of publication	Concepts described in the paper	Findings from the paper
10	Android REST APIs: Volley vs Retrofit 2018 International Symposium on Advanced Electrical and Communication Technologies, M. Lachgar, H. Benouda and S. Elfirdoussi	Presents a comparative study between Volley and Retrofit libraries,in order to disclose the advantages and limitations of each of them.	Using volley is more efficient than Retrofit for the specific android application

- **Paper Title:** Metrics for evaluating database selection techniques

  Year: 1999

  Author: J. C. French and A. L. Powell

  The increasing availability of online databases and other information resources in digital libraries has created the need for efficient and effective algorithms for selecting databases to search. A number of techniques have been proposed for query routing or database selection. We have developed a methodology and metrics that can be used to directly compare competing techniques. They can also be used to isolate factors that influence the performance of these techniques so that we can better understand performance issues. We describe the methodology we have used to examine the performance of database selection algorithms such as gGIOSS and CORI. In addition we develop the theory behind a "random" database selection algorithm and show how it can be used to help analyze the behavior of realistic database selection algorithms.

- **Paper Title:** Comparative analysis of the laravel and codeigniter frameworks: For the implementation of the management system of merit and opposition competitions in the State University Pennsula de Santa Elena

Year: 2018

Author: R. Valarezo and T. Guarda

The analysis was made of the comparative frameworks PHP: Laravel and CodeIgniter to improve productivity in the development of the system of management of competitions and merit and opposition in the University. It is recorded and analyzed the results of the tests performed on the two systems developed on the basis of the parameters and indicators established, in order to determine the framework with better productivity. The tools used in hardware: a computer, and in software - the frameworks PHP: Laravel and CodeIgniter, HTML 5, JavaScript, Bootstrap 3.3.6, Microsoft Visio 2010, JMeter and Quick Line Counter QLC. According to the obtained results it is inferred that the framework Laravel improves productivity in the development of the system of management of competitions and merit and opposition to the University, complying with the 100 percent of the parameters established in the analysis the same that is greater, compared to 68.86 percent of compliance with the Framework CodeIgniter. Was developed with the Framework Laravel the management system of competitions and merit and opposition in the University, because it provides better productivity in the development of web applications. It is recommended that the institution to make use of the system due to this web application automate all processes in an efficient manner.

- **Paper Title:** Mashup service release based on SOAP and REST

  Year: 2011

  Author: Huijie Su, Bo Cheng, Tong Wu and Xiaofeng Li

  In recent years,with more and more Mashup services appering, this paper proposed the solution to Mashup service release based on SOAP and REST. This paper introduces the Mashup service architecture of three layer system, dis-

cusses the function of each layer, and supports the rapid generation of Mashup services and service management. At the same time, in this architecture the paper introducts the details of the SOAP and the REST Web service design and implementation, completes the registration, operation, updation and delete of the Mashup service in a variety of ways, and enhances the compatibility.

- **Paper Title:** Research and implementation of Web Services in Android network communication framework Volley

  Year: 2014

  Author: Y. Shulin and H. Jieping

  The combination of Web Services and mobile devices will promote the development of mobile applications. Volley framework Google 2013 proposed has the advantages of convenient use and network request faster, but it does not support Web Ser vices. Extension of Volley, to support the Web Services, which ca n facilitate the Web Services application development, but also ca n improve the access performance of Web Services. On the basis of analysis and research of the Volley, Ksoap2 and Java Web Ser vices, through the implementation of the HttpStack interface and the expansion of Json Object Request to realize support for Web Services. The scheme uses JSON format to transfer data, support SSL/TLS protocol requests, custom parameter, sets or gets the request header. This scheme is good compatibility, easy to use, suitable for application on Android platform.

- **Paper Title:** A qualitative analysis of the performance of MongoDB vs MySQL database based on insertion and retriewal operations using a web/android application to explore load balancing Sharding in MongoDB and its advantages

  Year: 2017

Author: M. M. Patil, A. Hanni, C. H. Tejeshwar and P. Patil

Our world has evolved to an optimal point of advancement. The extravagant growth has helped in the invention of technologies, industry standards, gadgets, and devices that produce enormous amount of data that all require an essential data management and manipulation system. The data acquired from the various input and output sources are indulged in providing a certain infrastructure are also susceptible to damages if not treated well which may result in loss of data. To overcome this loss, various strategies that run parallel to prevent such loss are being used, one such example is the NoSQL MongoDB. MongoDb is a cross-platform, document oriented database that provides, high performance and easy scalability ensuring effective data management with its prominent feature of auto sharding. Sharding splits the database across multiple servers, increasing the capacity and scalability as required. This feature handles distribution of data in different nodes to maximize disk space and dynamically load balance queries. Partitioning the databases appropriately is a major step that determines the efficiency of sharding. This involves choosing an index of the MongoDB, competently as a shared key for further horizontal scaling of the database. Our current research involves the study of this load balancer. This paper intends to ascertain the need for NoSQL databases in the present situation and emphasize advancement of document-oriented database - MongoDB in particular by describing with a quantitative example that SQL databases are prone to deterioration when data is over loaded and MongoDB comes with inbuilt load balancer which makes it a better solution in applications with high data load. We describe the technology of sharding - auto load balancing feature of MongoDB and hope to provide a comprehensive insight of the process.

- **Paper Title:** A comparative study: MongoDB vs. MySQL

Year: 2015

Author: C. Gyrdi, R. Gyrdi, G. Pecherle and A. Olah

In this paper we will try to present a comparative study of non-relational databases and relational databases. We mainly focus our presentation on one implementation of the NoSQL database technology, namely MongoDB, and make a comparison with another implementation of relational databases, namely MySQL, and thus justifying why MongoDB is more efficient than MySQL. We will also present the advantages of using a non-relational database compared to a relational database, integrated in a forum in the field of personal and professional development. The NoSQL database used to develop the forum is MongoDB, and was chosen from a variety of non-relational databases, thanks to some aspects that we will highlight in this article. The database integration in the framework will also be presented.

- **Paper Title:** MySQL and NoSQL database comparison for IoT application

  Year: 2016

  Author: S. Rautmare and D. M. Bhalerao

  Internet of Things (IoT) concept has been around in tech world for few years now. IoT focuses on connection of number of smart devices. In near future, IoT will have applications in various domains and these applications are going to produce tremendous amount of data. With the continuous generation of heterogeneous data, problem arises to store, transfer & manage the data efficiently. Traditional database systems used Structured Query Language (SQL) database which has supported all the user requirements along with simplicity, robustness, flexibility, scalability, performance. But the main limitation they are facing is their static schema which is making RDBMS not suitable for IoT applications. On the other hand, NoSQL databases emerging in market have

claimed to perform better than SQL database. The NoSQL databases are non-relational, schema free, no joins, easy replication support, horizontally scalable, etc. Does NoSQL perform better than SQL in all application scenarios? An effort to answer the same has been made in this paper. This paper compares SQL and NoSQL databases for a small scale IoT application of water sprinkler system and investigates whether NoSQL performs better than SQL in different scenarios.

- **Title:** Evaluating web development frameworks: Django, Ruby on Rails and CakePHP.

  Year: 2009

  Author: Plekhanova, J.

  Web frameworks provide a golden mean between building an application from scratch and using an out-of-the-box content management system (CMS). This report focuses on three leading open source web development frameworks: Django, Ruby on Rails and CakePHP. All three frameworks have similar architectures and claim to have similar characteristics, such as greatly enhanced productivity and code re-use. This report provides a methodology to evaluate each framework. The methodology, criteria, and weights provided in this report are generic and comprehensive. Each organization should adapt the methodology of this report to its own unique context.

- **Title:** Python web development with Django. Addison-Wesley Professional.

  Year: 2008

  Author: Forcier, J., Bissex, P. and Chun, W.J.

  Learn Django A to Z

- **Paper Title:** A Method of Optimizing Django Based on Greedy Strategy.

Year: 2013

Author: J. Chou, L. Chen, H. Ding, J. Tu and B. Xu

Django on Python does well in agile web development. Python is dynamic programming language, as is known, its runtime efficiency is low. the question is how it will affect Django's efficiency. This Paper answered the question from the angle of memory optimization, by improving the efficiency of Python in memory use, we will see how it will affect django. Python manages non-container objects with the technology of memory pool, after a consequence of allocate and free operations, it will produce memory fragment, and the garbage collection module can hardly handle it. This paper proposed a new non-container object management greedy algorithm, it can minimize the probability of generating fragments, thus the garbage collection module can collect garbage non-container objects in time. We conduct our experiment on the benchmark of an open-source project named Unladen-Swallow. When Django renders a template, it will save about 10 percent memory, with almost the same time consuming.

- **Title:**REST Services with Django. In: Beginning Django.

  Year: 2017

  Author: Rubio D.

  Discover the Django web application framework and get started building Python-based web applications. This book takes you from the basics of Django all the way through to cutting-edge topics such as creating RESTful applications. Beginning Django also covers ancillary, but essential, development topics, including configuration settings, static resource management, logging, debugging, and email. Along with material on data access with SQL queries, youll have all you need to get up and running with Django 1.11 LTS, which is compatible with Python 2 and Python 3. Once youve built your web application, youll need to

be the admin, so the next part of the book covers how to enforce permission management with users and groups. This technique allows you to restrict access to URLs and content, giving you total control of your data. In addition, youll work with and customize the Django admin site, which provides access to a Django projects data.

- **Paper Title:** Designing Large Scale REST APIs Based on REST Chart

  Year: 2015

  Author: L. Li and W. Chou

  REST Chart is a Petri-Net based XML modeling framework for REST API. This paper presents two important enhancements and extensions to REST Chart modeling - Hyperlink Decoration and Hierarchical REST Chart. In particular, the proposed Hyperlink Decoration decomposes resource connections from resource representation, such that hyperlinks can be defined independently of schemas. This allows a Navigation-First Design by which the important global connections of a REST API can be designed first and reused before the local resource representations are implemented and specified. Hierarchical REST Chart is a powerful mechanism to rapidly decompose and extend a REST API in several dimensions based on Hyperlink Decoration. These new mechanisms can be used to manage the complexities in large scale REST APIs that undergo frequent changes as in some large scale open source development projects. This paper shows that these new capabilities can fit nicely in the REST Chart XML with very minor syntax changes. These enhancements to REST Chart are applied successfully in designing and verifying REST APIs for software-defined-networking (SDN) and Cloud computing.

- **Paper Title:** Design Patterns and Extensibility of REST API for Networking

Applications

Year: 2016

Author: L. Li, W. Chou, W. Zhou and M. Luo

REST architectural style has become a prevalent choice for distributed resources, such as the northbound API of software-defined networking (SDN). As services often undergo frequent changes and updates, the corresponding REST APIs need to change and update accordingly. To allow REST APIs to change and evolve without breaking its clients, a REST API can be designed to facilitate hypertext-driven navigation and its related mechanisms to deal with structure changes in the API. This paper addresses the issues in hypertext-driven navigation in REST APIs from three aspects. First, we present REST Chart, a Petri-Net-based REST service description framework and language to design extensible REST APIs, and it is applied to cope with the rapid evolution of SDN northbound APIs. Second, we describe some important design patterns, such as backtracking and generator, within the REST Chart framework to navigate through large scale APIs in the RESTful architecture. Third, we present a client side differential cache mechanism to reduce the overhead of hypertext-driven navigation, addressing a major issue that affects the application of REST API. The proposed approach is applied to applications in SDN, which is integrated with a generalized SDN controller, SOX. The benefits of the proposed approach are verified in different conditions. Experimental results on SDN applications show that on average, the proposed cache mechanism reduces the overhead of using the hypertext-driven REST API by 66 percent, while fully maintaining the desired flexibility and extensibility of the REST API.

- **Paper Title:** Android REST APIs: Volley vs Retrofit

  Year: 2018

Author: M. Lachgar, H. Benouda and S. Elfirdoussi

In the old days, networking in Android was a nightmare. Nowadays the problem is to find out which solution fits better the project's necessities. Therefore, almost every Android applications use a Web REST API for data transfer. Previously, developers preferred to write their own code to retrieve and analyze data. However, they now have a whole range of REST client libraries, which can speed up development by reducing the coding efforts, and efficiently analyze data. The REST (Representational State Transfer) has become the most commonly used way for creating, publishing, and consuming Web services, exploiting JavaScript Object Notation (JSON) as a data exchange format. Android Volley and Retrofit are the most used libraries for accessing the REST Web APIs today. In this paper, we will present a comparative study between these two libraries, in order to disclose the advantages and limitations of each of them.

- **Paper Title:** Research and implementation of Web Services in Android network communication framework Volley

  Year: 2014

  Author: Y. Shulin and H. Jieping

  The combination of Web Services and mobile devices will promote the development of mobile applications. Volley framework Google 2013 proposed has the advantages of convenient use and network request faster, but it does not support Web Ser vices. Extension of Volley, to support the Web Services, which ca n facilitate the Web Services application development, but also ca n improve the access performance of Web Services. On the basis of analysis and research of the Volley, Ksoap2 and Java Web Ser vices, through the implementation of the HttpStack interface and the expansion of Json Object Request to realize support for Web Services. The scheme uses JSON format to transfer data, sup-

port SSL/TLS protocol requests, custom parameter, sets or gets the request header. This scheme is good compatibility, easy to use, suitable for application on Android platform.

Chapter 3

SOFTWARE REQUIREMENT SPECIFICATION

3.1 Introduction

3.1.1 Purpose

The existing architecture for storage and access of ASTROSAT data is rudimentary and only involves HTML and data-linking as of now for managing the huge amount of quotidian data generated. The existing structure lacks the usage of any form of structural databases or relational databases for data retrieval and storage. The new architecture suggests the use of a structured database for storing and managing the data through a web service and availing the data to the end user in an android application. This architecture proposes improved and ubiquitous data access and reduced latency. It will help in Portability and Improved Management of ASTROSAT CZTI Data. Optimizing the storage solution of CZTI data by implementing web services, database servers and front end android interface. The web service will extract data from the existing servers via data migration streams and store the data in a specific format that will optimize the retrieval and access of the satellite data and reduce the latency.

- **Storage:** The scope of this project involves facilitating the storage of the persis-

tent data generated and transmitted by the ASTROSAT (CZTI) by the virtue of features provided by a RDBMS, typically MySQL. The storage facility will act as a data warehouse for the DQR of the ASTROSAT and will support the dynamic access and retrieval of the required datasets.

- **Retrieval**: Retrieval of the data is made possible on portable devices like mobile phones by firing the appropriate queries on the tables created using the RDBMS. The database schema is designed to improve the access and retrieval speed of the data. The search can be made on the required attributes as the storage schema involves composite primary key to generate a unique identification for every orbit to implement exclusivity in the database.

- **Data Migration:** The project will provide migration adequate data migration services to migrate the existing data that is stored on the IUCAA servers via Poorva & Rohini to the new RDBMS server using appropriate data migration facilities. The data migration allows replication of the existing ASTROSAT dataset to the new database server created for aiding faster and safer retrieval of the data. The new rows added everyday to the IUCAA servers need to be migrated to the RDBMS server in real time to provide instant portable access and hence the data migration services are essential.

- **Android Application:** The android application is being developed to allow mobile access to all the Data Quality Reports anytime, anywhere. The UI will allow easier interpretation of the DQR and intuitive interface with the data storage architecture that will act as the back end for the same.

- **Web Service:** The web service acts as a middle bridge or link between the RDBMS server and the mobile application. It receives the request from the application for the retrieval of the data and fires the required queries on the

database server. The data is retrieved from the appropriate tables and sent to the android application and displayed to the user. The web service is also required to initiate the migration services as soon as the new data of the orbits is received through the ASTROSAT.

3.1.2 Software Development Life Cycle Model

Spiral Model

In this system, We have used Spiral Model. The spiral model combines the idea of iterative development with the systematic, controlled aspects of the waterfall model. This Spiral model is a combination of iterative development process model and sequential linear development model i.e. the waterfall model with a very high emphasis on risk analysis. It allows incremental releases of the product or incremental refinement through each iteration around the spiral.

Spiral Model Design

The spiral model has four phases. A software project repeatedly passes through these phases in iterations called Spirals.

- **Identification:** This phase starts with gathering the business requirements in the baseline spiral. In the subsequent spirals as the product matures, identification of system requirements, subsystem requirements and unit requirements are all done in this phase.

  This phase also includes understanding the system requirements by continuous communication between the customer and the system analyst. At the end of the spiral, the product is deployed in the identified market.

- **Design:** The Design phase starts with the conceptual design in the baseline spiral and involves architectural design, logical design of modules, physical product

design and the final design in the subsequent spirals.

- **Construct or Build:** The Construct phase refers to production of the actual software product at every spiral. In the baseline spiral, when the product is just thought of and the design is being developed a POC (Proof of Concept) is developed in this phase to get customer feedback.

  Then in the subsequent spirals with higher clarity on requirements and design details a working model of the software called build is produced with a version number. These builds are sent to the customer for feedback.

- **Evaluation and Risk Analysis:** Risk Analysis includes identifying, estimating and monitoring the technical feasibility and management risks, such as schedule slippage and cost overrun. After testing the build, at the end of first iteration, the customer evaluates the software and provides feedback.

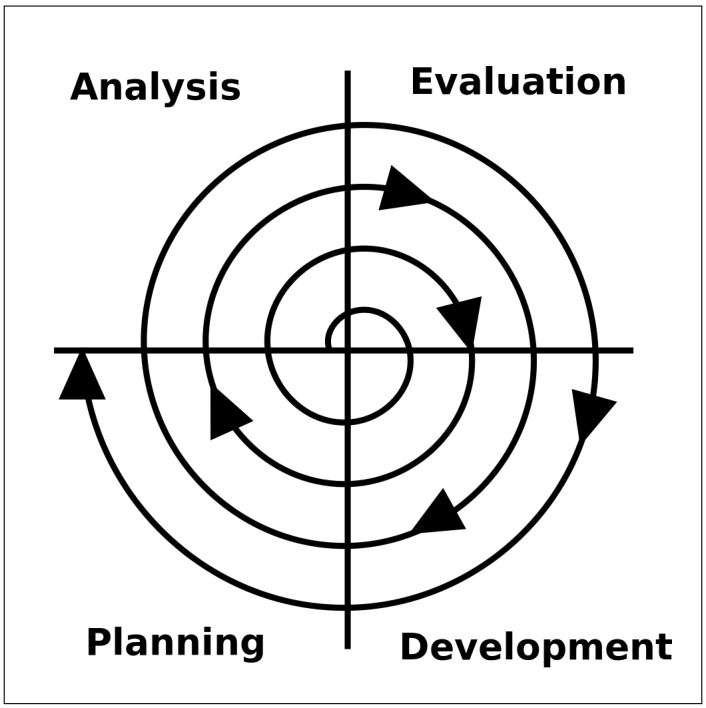


Figure 3.1: Spiral Model

3.1.3 User Class and Characteristics

In this system, the 2 main entities are:

- **Administrator:** This user class includes the people responsible for establishing and maintaining the overall software and hardware involved in the project. The administrators are required to ensure the irresolute functioning of the software without intermittent issues of any kind.
- **End User:** This class of users is the one accessing the DQR on their mobile devices or computers. This user class is unbeknownst of the internal architecture of the ASTROSAT data repository as the software creates an abstraction between the public user class and the software internals. This user class can access the data anytime and anywhere and perform certain CRUD (limited to a specified extent) operations depending upon their requirements.

Assumptions and Dependencies:

- Data required by the system is pre-processed. Filtered to remove Noisy Data from ASTROSAT reading.
- Data is in ASCII Format. Primary reading by ASTROSAT is in FITS format, the conversion from FITS to ASCII should be done.
- Data is suitable for RDBMS like MySQL.
- Data in the Databases is error free, non redundant, consistent.
- Data should follow the 5Vs Velocity, Value, Variety, Volume, Variability.
- Web service will perform without failure and maximum up time.
- Android app will not crash and perform efficiently.
- Django framework is used.

- No Database Schema conflict with each other.

3.1.4 Functional Requirements

Data Description Flexible Image Transport System (FITS) is an open standard defining a digital file format useful for storage, transmission and processing of data: formatted as N-dimensional arrays (for example a 2D image), or tables. FITS is the most commonly used digital file format in astronomy. The FITS standard has special (optional) features for scientific data, for example it includes many provisions for describing photo-metric and spatial calibration information, together with image origin metadata.

- **Business Rules:** End Users cannot be allowed access to background database. End Users cannot edit or modify application metrics and settings. The implemented Django based Web Service should not interfere with incoming Server Data in any scenario. The data to be displayed on the android app has to be appropriately managed so that there is zero loss in data quality and information.

- **Authentication:** Since the system needs to handle a lot of research specific ASTROSAT data, a server authentication function is essential for ensuring security. Users who do not have the correct access rights will be prevented from connecting to the database. There are two groups of users of the system with different access rights: Administrator - acts as the system administrator and can perform all functions including view, edit and modify information; End User - can view and share information which is displayed and updated periodically on the android application;

- **Administrative Functions:** This functional requirement includes the people responsible for establishing and maintaining the overall software and hardware

involved in the project. The administrators are required to ensure the irresolute functioning of the software without intermittent issues of any kind which include unethical or unauthorized use of data, manhandling of valuable project resources and interference with the functioning of the main data Server..

- **Audit Tracking:** The system should allow authorized users to: View or create a report of all newly created records View or create a report of all user access profiles View or create a report of all user id login and logout times over a specific period View or create a report of functional usage by user ID of system activity over a specific period, e.g. list the number of times each type of system activity (report, query, accession, etc.) was accessed on a certain day by a user View or create a report by system activity on user access over a specific period, e.g. for each system activity (report, query, accession, etc.) list each user who accessed on a particular day View or create a report of the queries performed by users.

- **External Interface:** The overall style should be built up easily in order for end users to use it easily and efficiently. After accessing the information through the android application, the users should feel comfortable while looking at it and browsing through it.

  The tabular design of the information represented should not be changed and should be replicated in a more refined manner.

  The ASTROSAT logo and the current basic design of the portal should be displayed. The system should be attractive and user friendly. The design and the color should make users feel comfortable when using the system instead of flashing useless colors on the screen.

- **Certification Requirements:** No special certification requirements are re-

quired.

- **Reporting Requirements:** All reporting requirements have been covered under Audit Tracking.

- **IP and Security Requirements:** All existing copyright and security requirements apply as they are to the application and architectural framework being developed.

3.2 Non Functional Requirements

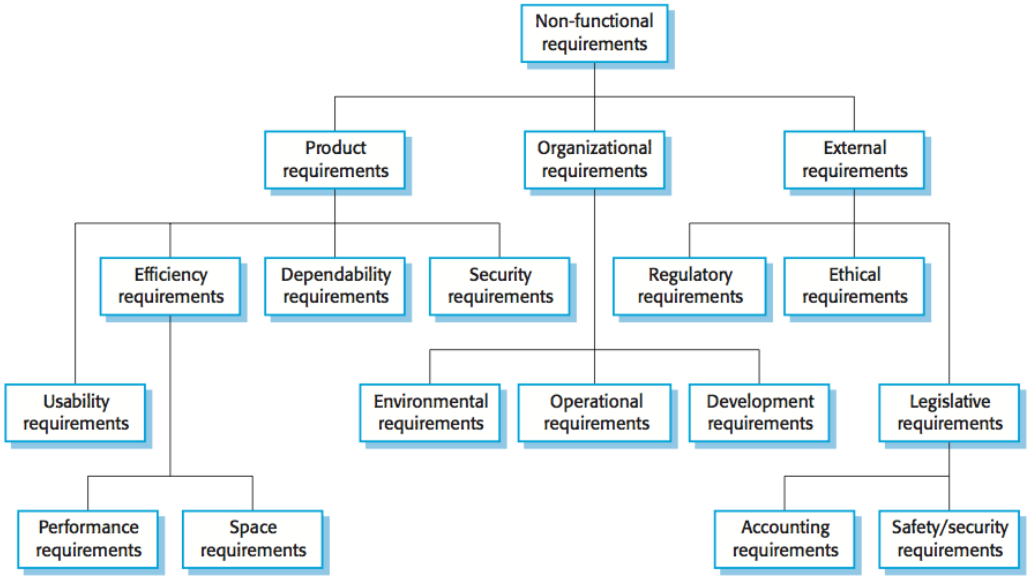


Figure 3.2: Non Functional Requirement

Performance:

The existing architecture is designed based on storage of DQR reports in plain HTML format after retrieving them from IUCAA servers which when accessed through the existing website platform is achieved vis a vis a lot of latency issues with the entire home page sometimes taking more than 2 minutes to load on a stable internet connection.

Our objective as discussed is to reduce this response time significantly (Less than 1.08.231 minutes at 1.4 Mbps stable connection) while accessing the data on mobile devices since the current architectural model is not feature based and is not able to support the large datasets generated on a daily basis efficiently. The proposed mobile application would interact with the generated database entries and the response time would go down much to the helpful usability of the end users.

Scalability:

The underlying Django based architecture and the front end Android application shall be scalable to accommodate unrestricted growth in the number of Data Quality Reports for the orbits being added with the added abilities to cope with increasing processing load, expanding UI characteristics and identifying and removing possible causes for degradation which are currently absent in the existing model. (The scale of end users accessing data and viewing it will not restrict growth or negatively affect application performance.)

Availability:

Routine software changes shall be applied no more frequently than once every three months, and whenever possible shall be installed while the ASTROSAT-CZTI DQR application is active which will work towards reducing downtime impact on the usage, partial availability impact on the accessibility and also minimizing complete unavailability.

Reliability:

The architectural model being proposed along with the android application will be developed with an aim to minimise the probability of failure on demand which shall remain 0.001 (1 out of 1000 requests) when an end user requests

data viewing on the mobile application. The possible causes of failures may include unstable network connection (rectified by a check on application start) or unavailability of data from IUCAA servers (rectified by a toast or display prompt).

Survivability:

This non-functional requirement addresses is the extent to which the android application and underlying web service continues to function and recovers in the presence of a system failure. The system shall use various failure detection techniques and fault recovery techniques. One of the probable functional scenarios would be that When an update failure is detected all updates performed during the failed session shall be rolled back to restore the data to pre-session condition and then the app would function normally as before.

Capacity:

The system should be able to manage all the information incoming from the database efficiently with optimal response time and high throughput value.

Data Integrity:

Integrity requirements address the user concern for the accuracy and authenticity of the data. Possible practices include needs regarding data restore procedures, and authenticity of data displayed on the Android app with respect to the original data source which can be verified once every week by a Administrator responsible for the same.

Environmental Requirements:

The system is Web based therefore it will be used in any environment that allows Web access.

Usability:

Usability is the ease with which the user is able to learn, operate, prepare inputs and interpret outputs through interaction with the Android Application. The developed android app needs to fulfill the ease of entry, ease of viewing, ease of handling and possible likability issues concerning the End User. One of the proposed features regarding the same includes the option to share the required DQR data with other users/non-users through third party applications.

Adaptability:

The android application should be compatible with all major version and screen standards so via all major network providers and with necessary scale and performance upgrades be adaptive.

Security:

Security requirement is the extent to which the developed architecture along with the android application is safeguarded against deliberate and intrusive faults from internal and external sources. Since the application has only one primary functionality i.e. viewing DQRs, the architectural security standards will be applied to the database ports and web service authentication access.

Confidentiality:

Since the ASTROSAT-CZTI DQR research oriented data is to be made publicly available, privacy is not a concern till the time user specific data access upgrades are not made.

Supportability

The system should need to be entirely self-supporting since the users would be using it only to view and share data.

3.2.1 External Interface Requirements

- **User Interface:**

  Android devices come in all shapes and sizes, so your app's layout needs to be flexible. That is, instead of defining your layout with rigid dimensions that assume a certain screen size and aspect ratio, your layout should gracefully respond to different screen sizes and orientations. By supporting as many screens as possible, your app can be made available to the greatest number of users with different devices, using a single APK. Additionally, making your app flexible for different screen sizes ensures that your app can handle window configuration changes on the device, such as when the user enables multi-window mode.

- **Xml language:**

  Android provides a straightforward XML vocabulary that corresponds to the View classes and sub classes, such as those for widgets and layouts. Declaring your UI in XML allows to separate the presentation of the app from the code that controls its behavior. Using XML files also makes it easy to provide different layouts for different screen sizes and orientations (discussed further in Supporting Different Screen Sizes). The Android framework gives the flexibility to use either or both of these methods to build the app's UI. For example, one can declare app's default layouts in XML, and then modify the layout at runtime.

- **Java xml interfacing:**

  The Java API for XML Processing (JAXP) is for processing XML data using applications written in the Java programming language. JAXP leverages the parser standards Simple API for XML Parsing (SAX) and Doc-

ument Object Model (DOM) so that you can choose to parse your data as a stream of events or to build an object representation of it. JAXP also supports the Extensible Stylesheet Language Transformations (XSLT) standard, giving you control over the presentation of the data and enabling you to convert the data to other XML documents or to other formats, such as HTML. JAXP also provides namespace support, allowing you to work with DTDs that might otherwise have naming conflicts.

- **Java XML interfacing:**

  The Java API for XML Processing (JAXP) is for processing XML data using applications written in the Java programming language. JAXP leverages the parser standards Simple API for XML Parsing (SAX) and Document Object Model (DOM) so that you can choose to parse your data as a stream of events or to build an object representation of it. JAXP also supports the Extensible Stylesheet Language Transformations (XSLT) standard, giving you control over the presentation of the data and enabling you to convert the data to other XML documents or to other formats, such as HTML. JAXP also provides namespace support, allowing you to work with DTDs that might otherwise have naming conflicts.

- **Android App features:**

  Intro Slider: Android Introduction Slider is basically used to introduce the major features of the application to the user. It also attracts and helps the user to know about the app. It is also called android welcome screen and is used to guide the user on how to use the app just after installation. It is used in almost all popular apps and games these days.

  Navigation drawer: The navigation drawer is a UI panel that shows your app's main navigation menu. It is hidden when not in use, but appears

when the user swipes a finger from the left edge of the screen or, when at the top level of the app, the user touches the drawer icon in the app bar.

Fragments: Fragment represents a behavior or a portion of user interface in a Fragment Activity. You can combine multiple fragments in a single activity to build a multi-pane UI and reuse a fragment in multiple activities. You can think of a fragment as a modular section of an activity, which has its own life cycle, receives its own input events, and which you can add or remove while the activity is running (sort of like a "sub activity" that you can reuse in different activities).

Tab layout: Tab layout are visible below toolbar with view pager, used to create swipe able views on. Tabs are designed to work with fragments. Use them to swipe fragments in view pager. In this article, we are going to show you how to implement material design tabs in your android app.

Table layout: Table Layout positions its children into rows and columns. Table Layout containers do not display border lines for their rows, columns, or cells. The table will have as many columns as the row with the most cells. A table can leave cells empty. Cells can span multiple columns, as they can in HTML.

Graph view library: Graph View is a library for Android to programmatically create flexible and nice-looking diagrams. It is easy to understand, to integrate and to customize. Create Line Graphs, Bar Graphs, Point Graphs or implement your own custom types.

3.2.2 Hardware Interface

- **Smartphone**

  The system will interact with the humans through an Android Smartphone.

The Android smartphone will act as a link between the web service and the user, where the user can request for the data and the appropriate results will be displayed to the user after retrieval from the database. The smartphone will enable the user to access the data anytime and anywhere without requiring him/her to use a very high capability computing device. The android application is being developed to allow mobile access to all the Data Quality Reports anytime, anywhere. The UI will allow easier interpretation of the DQR and intuitive interface with the data storage architecture that will act as the back end for the same. The user can view the textual and graphical data (in the form of images) on the smartphone device and perform operations like searching of the required data using several attributes of the tables.

- **Workstation**

  The computer workstation will enable the administrator class of users to perform the administrative and maintenance related operations on the system. The data access can also be made possible on a computer through a web application through certain modifications of the existing system. General public need not use computers to access the ASTROSAT DQRs. The computer is an essential external interface to establish and troubleshoot the system if necessary.

3.2.3 Software Interfaces

- **Operating System: Android** Android is a Linux based operating system it is designed primarily for touch screen mobile devices such as smartphones and tablet computers. The operating system have developed a lot in last 15 years starting from black and white phones to recent smartphones or mini

computers. One of the most widely used mobile OS these days is android. The android is a software that was founded in Palo Alto of California in 2003. The android is a powerful operating system and it supports large number of applications in Smartphones. These applications are more comfortable and advanced for the users. The hardware that supports android software is based on ARM architecture platform. The android is an open source operating system means that its free and any one can use it. The android has got millions of apps available that can help you managing your life one or other way and it is available low cost in market at that reasons android is very popular.

- **Linux Kernel**

  Android was created on the open source kernel of Linux. One main reason for choosing this kernel was that it provided proven core features on which to develop the Android operating system. The features of Linux kernel are:

  1. Security: The Linux kernel handles the security between the application and the system.
  2. Memory Management: It efficiently handles the memory management thereby providing the freedom to develop our apps.
  3. Process Management: It manages the process well, allocates resources to processes whenever they need them.
  4. Network Stack: It effectively handles the network communication.
  5. Driver Model: It ensures that the application works. Hardware manufacturers can build their drivers into the Linux build.

- **Libraries**

  Running on the top of the kernel, the Android framework was developed

with various features. It consists of various C/C++ core libraries with numerous of open source tools. Some of these are:

1. The Android runtime: The Android runtime consist of core libraries of Java and ART(the Android RunTime). Older versions of Android (4.x and earlier) had Dalvik runtime.
2. OpenGL(graphics library): This cross-language, cross-platform application program interface (API) is used to produce 2D and 3D computer graphics.
3. WebKit: This open source web browser engine provides all the functionality to display web content and to simplify page loading.
4. Media frameworks: These libraries allow you to play and record audio and video.
5. Secure Socket Layer (SSL): These libraries are there for Internet security.

- **Android Runtime**

  newline It is the third section of the architecture. It provides one of the key components which is called Dalvik Virtual Machine. It acts like Java Virtual Machine which is designed specially for Android. Android uses its own custom VM designed to ensure that multiple instances run efficiently on a single device. The Dalvik VM uses the devices underlying Linux kernel to handle low-level functionality, including security, threading and memory management.

- **Application Framework**

  The Android team has built on a known set proven libraries, built in the background, and all of it these is exposed through Android interfaces.

These interfaces warp up all the various libraries and make them useful for the Developer. They dont have to build any of the functionality provided by the android. Some of these interfaces include:

1. Activity Manager: It manages the activity lifecycle and the activity stack.
2. Telephony Manager: It provides access to telephony services as related subscriber information, such as phone numbers.
3. View System: It builds the user interface by handling the views and layouts.
4. Location manager: It finds the devices geographic location

3.2.4 Communication Interfaces

- **Web services-Django framework:**

  Django is a high-level Python Web framework that encourages rapid development and clean pragmatic design. A Web framework is a set of components that provide a standard way to develop websites fast and easily. Djangos primary goal is to ease the creation of complex database-driven websites. Its free and open source.

- **Wireless internet protocol:**

  Wireless Internet Protocols are the suite of wireless protocols after Wireless Application Protocol 2.0 (WAP). It includes XHTML Basic, Nokia's XHTML Mobile Profile, and future developments of WAP by the Open Mobile Alliance.

  Wireless Internet Protocols are able to deliver XHTML pages to appropriate wireless devices without the need for HTTP to WAP proxies.

  Using Wireless Internet Protocols, web pages can be rendered differently in

web browsers and on handhelds without the need for two different versions of the same page.

- **CSMA/CA (Carrier-sense multiple access with collision avoidance):**

  CSMA/CA in computer networking, is a network multiple access method in which carrier sensing is used, but nodes attempt to avoid collisions by transmitting only when the channel is sensed to be "idle". When they do transmit, nodes transmit their packet data in its entirety.

- **MIMO (multiple-input and multiple-output):**

  Some standards, such as IEEE 802.11n and IEEE 802.11ac for Wi-Fi allow a device to have multiple antennas. Multiple antennas enable the equipment to focus on the far end device, reducing interference in other directions, and giving a stronger useful signal. This greatly increases range and network speed without exceeding the legal power limits. IEEE802.11n can more than double the range. Range also varies with frequency band. Wi-Fi in the 2.4 GHz frequency block has slightly better range than Wifi in the 5 GHz frequency block used by 802.11a (and optionally by 802.11n). Under optimal conditions, IEEE802.11ac can achieve communication rates of 1 Gbit/s.

- **Mobile data generations:**

  Mobile data generation refers to the change in nature of service compatible transmission technology and new frequency bands.

3.3 System Requirements

3.3.1 Database Requirements

- Data should be in structured format The data to be stored and accessed by the system should be strictly in structured format; Semi-structured or unstructured data is not supported. Suitable RDBMS is advised to allow optimum operation of the system.

- Images should be in jpeg or png Any other image format except jpeg and png cannot be stored or processed by the system. The system assumes a consistency amongst all the images and graphs in the database and requires them to be strictly in JPEG or PNG format only.

- Data should be normalized To facilitate speed up in the access of the database from the existing system, the newly created system requires the data to be stored and normalized. Redundancy, inconsistency and duplicate entries will hinder the efficient operation of the system.

3.3.2 Software Requirements

- **Ubuntu 16.04 64 bit**

  A 64 bit Open source Operating system, it is used by the developing community. It is a widely used Linux flavour.

- **Android Studio (version 3.2.1)**

  Android Studio is the official integrated development environment (IDE) for Google's Android operating system, built on JetBrains' IntelliJ IDEA software and designed specifically for Android development. It is available for download on Windows, macOS and Linux based operating systems. It is a replacement for

the Eclipse Android Development Tools (ADT) as the primary IDE for native Android application development.

- **MySQL**

  MySQL is an open-source relational database management system (RDBMS). It extends Cross-platform support. SQL provides very structured schema for easy database management.

- **Pycharm**

  PyCharm is an IDE developed by JetBrains. PyCharm provides smart code completion, code inspections, on-the-fly error highlighting and quick-fixes, along with automated code refactorings and rich navigation capabilities. Various features of PyCharm are VCS, Deployment and Remote Development,Debugging, Testing and Profiling,Database tools. In addition to Python, PyCharm provides first-class support for various Python web development frameworks, specific template languages, JavaScript, CoffeeScript, TypeScript, HTML/CSS, AngularJS, Node.js, and more.

- **Android OS on smartphone (5.0 minimum)**

  The largest and most ambitious release for Android.This release is packed with new features for users and thousands of new APIs for developers. It extends Android even further, from phones, tablets, and wearables, to TVs and cars.

- **Android Emulator**

  The emulator provides almost all of the capabilities of a real Android device. You can simulate incoming phone calls and text messages, specify the location of the device, simulate different network speeds, simulate rotation and other hardware sensors, access the Google Play Store, and much more. Testing your app on the emulator is in some ways faster and easier than doing so on a physical

device. For example, you can transfer data faster to the emulator than to a device connected over USB.

3.3.3 Hardware Requirements

- **Android smartphone**

  The system has an user interface built upon the Android OS platform as of now. Thus, to access the database and DQR the user is required to have an Android Smartphone. The app supports several android features and allows the user to access the data of the ASTROSAT remotely. The android smartphone should have minimum android 5.0 and RAM of minimum 2GB for the app to function. The app will require a stable and consistent data connection to communicate with the web service and retrieve data from the database server to display the appropriate records of the orbits to the user.

  Screens MUST be at least 2.5 inches in physical diagonal size Density MUST be at least 100 dpi The aspect ratio MUST be between 1.333 (4:3) and 1.779 (16:9) The display technology used consists of square pixels

Chapter 4

SYSTEM DESIGN

4.1 Existing System

ASTROSAT CZTI data is acquired from the satellite in the FITS format and stored after transforming it into ASCII and JPEG at IUCAA servers. The data captured by ASTROSAT for any orbit under observation is stored in the form of DQRs (Data Quality Reports) which consist of Observation Information, Noise Dominated Fractions, Top Noisy Pixels, Detector Plane Histograms etc.

The current storage of the DQR is in HTML (Hypertext Markup Language) format on the IUCAA web servers: it generates dynamic web pages for every new orbit added to the database. It takes infeasible time (approximately two minutes on a stable data connection) to load the webpage entirely and then perform the operations. The interface is quite rudimentary and does not support many features. Currently the data is not suitable to be accessed on mobile devices as mobile processing power is not sufficient to process such huge amounts of data. The ASTROSAT generates approximately 15 rows of astronomical data each day which further aggravates the problem of efficient data storage and access.

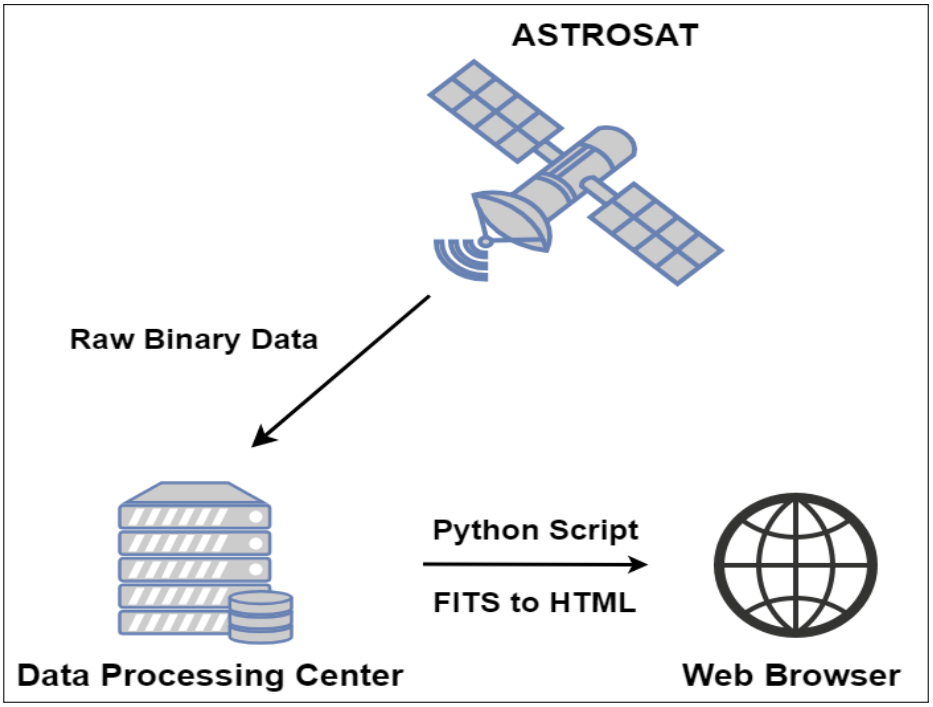


Figure 4.1: Existing system of ASTROSAT-CZTI Data Quality Reports

4.1.1 Current Scenario

The existing system for displaying ASTROSAT DQRs includes a HTML generation script in Python which transforms raw FITS data from the satellite into human readable HTML web pages. The major drawback of this system is that it lacks the presence of the a dedicated server to regulate and control the flow of such huge quantities of data. Now due to this flaw i.e. the absence of a dedicated server, it makes the website significantly heavy and also makes it difficult to filter and load required results relevant to the user. The architecture currently has negligible portability options which makes it difficult to access the DQRs on a mobile application level.

The above figure depicts the architecture currently in use. The data processing centre continuously runs the Python script which generates the HTML web pages from FITS data to be displayed on the web browser as shown in the figure.

4.2 Proposed System

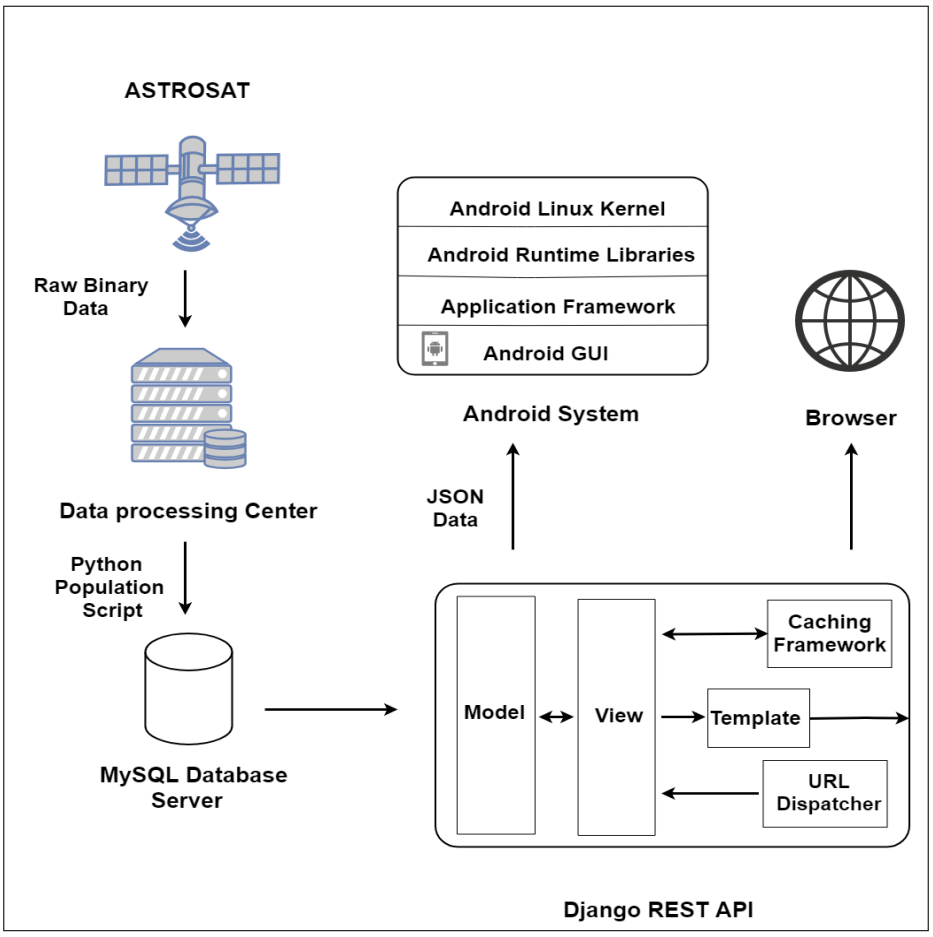


Figure 4.2: Proposed system for ASTROSAT-CZTI Data Quality Reports

4.2.1 ARJUN : The Application Server

ARJUN (Access and Retrieval Jointly Using N-tier architecture) is the proposed application server which will play a major role in overcoming the previously mentioned shortcomings.

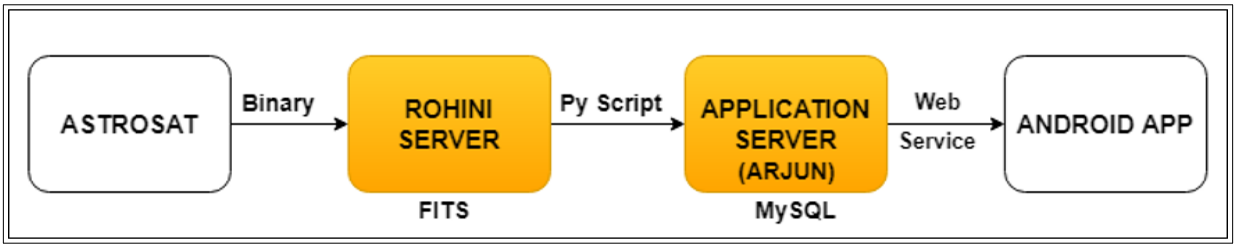


Figure 4.3: ARJUN : The Application Server

We propose a modified Python populating script which would fetch data from the Data Processing Centre and store in ARJUN. This would be done periodically as and when data is captured from ASTROSAT. This Python script would be scheduled in a CRON service in the system background (daemon) which would run automatically whenever database needs to be updated.

The design of the database used in ARJUN is highly structured, simplified, scalable and also facilitates easy and fast access. The database schema used is RDBMS (Relational Database Management System) based on MySQL. This storage facility will act as a data warehouse for the DQRs of the ASTROSAT and will support the dynamic access and retrieval of the required datasets. Retrieval of the data is made possible on portable devices like mobile phones by firing the appropriate queries on the tables created using the RDBMS. The search can be made on the required attributes as the storage schema involves composite primary key to generate a unique identification for every orbit to implement exclusivity in the database.

The Android application will communicate with ARJUN via an efficient Django REST Framework based API and fetch data accordingly.

4.2.2 Web Service

The web service acts as a middle bridge or link between the RDBMS server and the mobile application. It receives the request from the application for the retrieval of the data and fires the required queries on the database server. The data is retrieved from the appropriate tables and sent to the android application and displayed to the user. The web service is also required to initiate the migration services as soon as the new data of the orbits is received through the ASTROSAT. Web service makes the system available over the internet and uses a standardized messaging system. All communication is in standard format, therefore web services are not tied to any one

operating system or programming language. Thus makes the system cross platform.

The Web Service in our architecture consists of three main parts: Django Framework, REST API, Google Volley.

Django Framework

Django a high-level Python internet framework that encourages fast development and clean, pragmatic style. Our architecture makes use of the distributed system feature of Django which involves the creation of a separate "app" for each functionality of the application. Thus it helps in overall load balancing of our system. The Django framework has a Model View Template architecture.

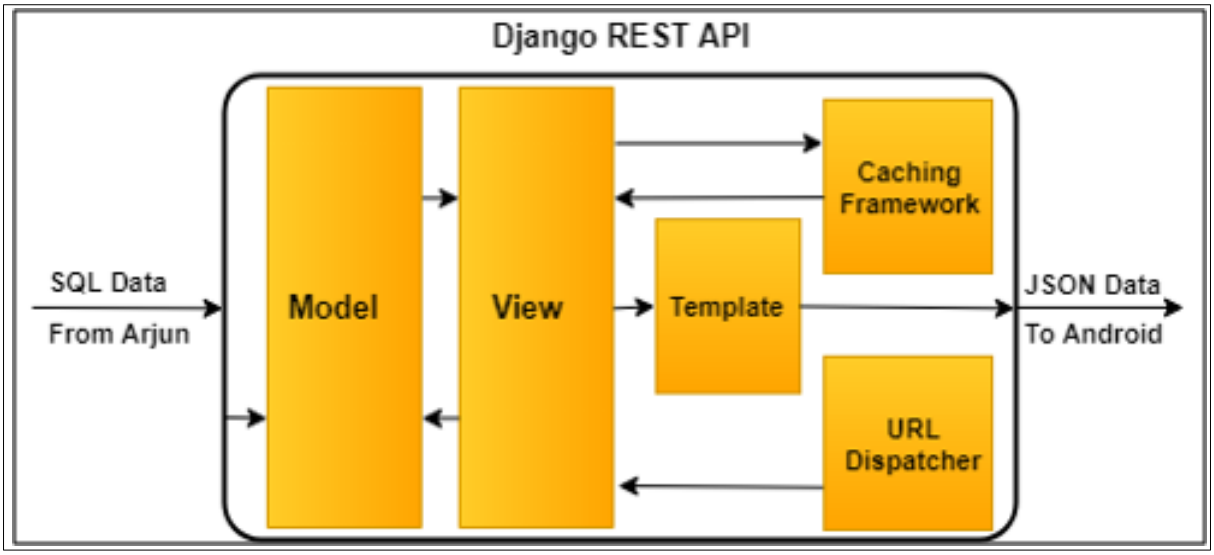


Figure 4.4: Django Architecture

- **Model** in Django is single source of information about our data. It contains the essential fields and behaviors of data. Each model maps to a single database table. This makes the system loosely coupled and highly cohesive. Modularity of

the architecture makes it highly efficient to handle large requests at a time.The underlying Django based architecture is scalable to accommodate unrestricted growth in the number of Data Quality Reports for the orbits being added with the added abilities to cope up with increasing processing load. It removes possible causes for degradation which are currently present in the existing system. The Model acts as a bridge between ARJUN and View of Django.

- **View** is a Python function that takes a online request and returns a response. This response can be HTML contents, JSON (Java Script Object Notation) Data etc. View controls the data requests and regulatory operations. Important functionalities like URL mapping, serializers and JSON generation are performed by a Django view. The JSON data will be used in combination with Google Volley for mobile based architecture. For web application the View uses HttpRequest and HttpResponse functions to render HTML pages.

- **Template** system helps in generating reusable dynamic HTML pages.Being an internet framework, Django requires a convenient functionality to generate hypertext markup language dynamically. The most common approach relies on templates. A template contains the static components of the required hypertext markup language output and as some special syntax describing how dynamic content is inserted [20]. In our MVT architecture, Model and View are the fractions that belong to the server side and Template is the component that belongs to the client side of the system. The proposed website for the ASTROSAT-CZTI DQRs consists of certain static HTML aspects that may remain unchanged throughout the website. Template helps us to identify and discern the static (HTML tags) and dynamic (data to be displayed) aspects of the website facilitating us in maintaining a lightweight framework as compared

to the existing architecture. Further Template is useful in separating the HTML and Python code of our system that allows the data retrieved from the ARJUN server to be displayed dynamically which is flanked by the static HTML tags. This has helped us in implementing data re-usability in our web framework.

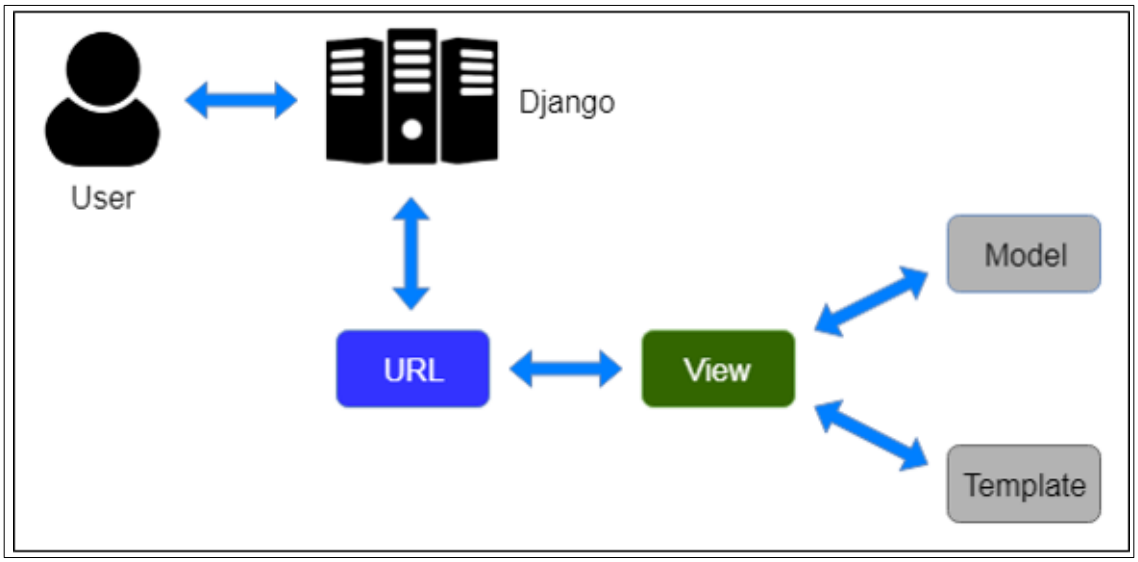


Figure 4.5: Model-View-Template

REST API

REST (Representational State Transfer) API (Application Programmable Interface) uses HTTP functions like GET, POST, DELETE, PUT etc [27]. In our RESTful API, endpoints (URLs) define the structure of the API and how end users access data from our application using the above mentioned HTTP methods. REST framework is used as a base for Serialization. Serializers implemented in our Django framework allow complex data such as query sets and model instances to be converted to native Python data types that can then be easily rendered into JSON, XML or other content types.

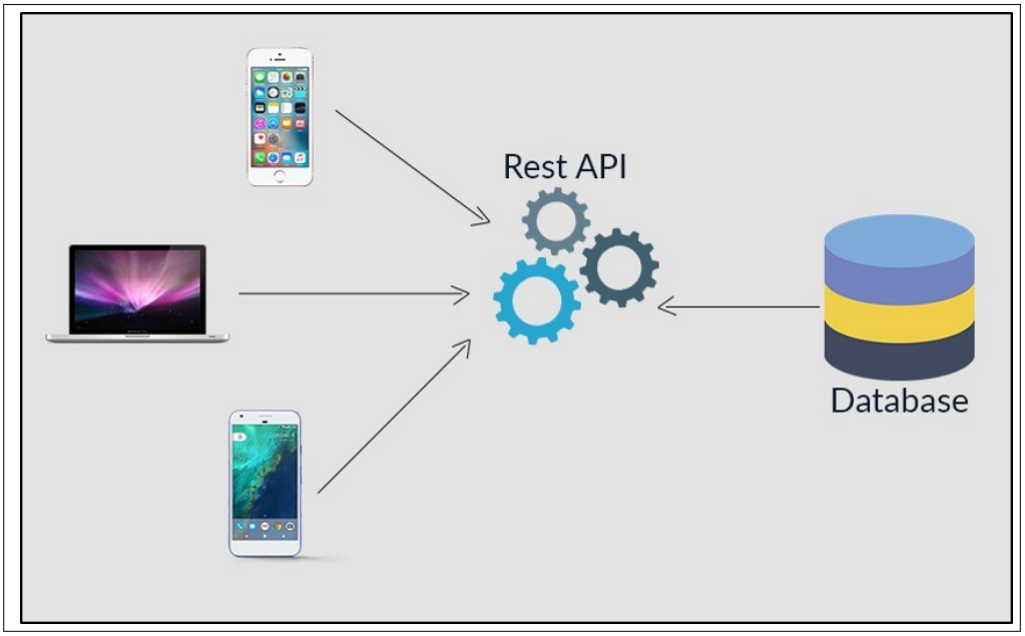


Figure 4.6: REST API

Volley

While researching about Android's AsyncTask for handling networking requests, we found out AsyncTask cant prioritize parallel/multiple requests in an orderly manner when it comes handling huge amount of data. Besides it contains a lot of boilerplate code.

Google Volley is a network library provided by Google exclusively for Android which helps in easier and faster retrieval of JSON objects from the REST API [6]. It has some powerful options and could be a ton quicker than AsyncTask. In our Android architecture, network requests in Volley are added to a RequestQueue. Instead of creating a new instance of RequestQueue every time, we follow the singleton pattern by creating a single instance of RequestQueue throughout our application. This results in faster JSON retrieval speed.

4.3 User Interface

The third important component of our architectural model includes generic methods to implement the interface of our system. These may include Linux based Android systems or Apple's iOS operating system. The desktop interface has no barriers and will work with equally great efficiency and is platform independent with support for every major OS and platform.

Figure 4.7: Existing Web Portal

Report sections: Obs info | Basic stats | L1 data integrity | Saturation | Noise dom frac | Top noisy pixels | DPH | Count rate + images | HK plots

Data Quality Report
Orbit 17393

Creation date: 2018-12-16T13:25:37.848642
Switch to: Orbit-wise | Merged OBSID-wise
web.py, SVN revision $Rev: 1121 $

Source files: Data quality report (detailed, human readable) | Data quality report (concise, machine readable)

^TOP

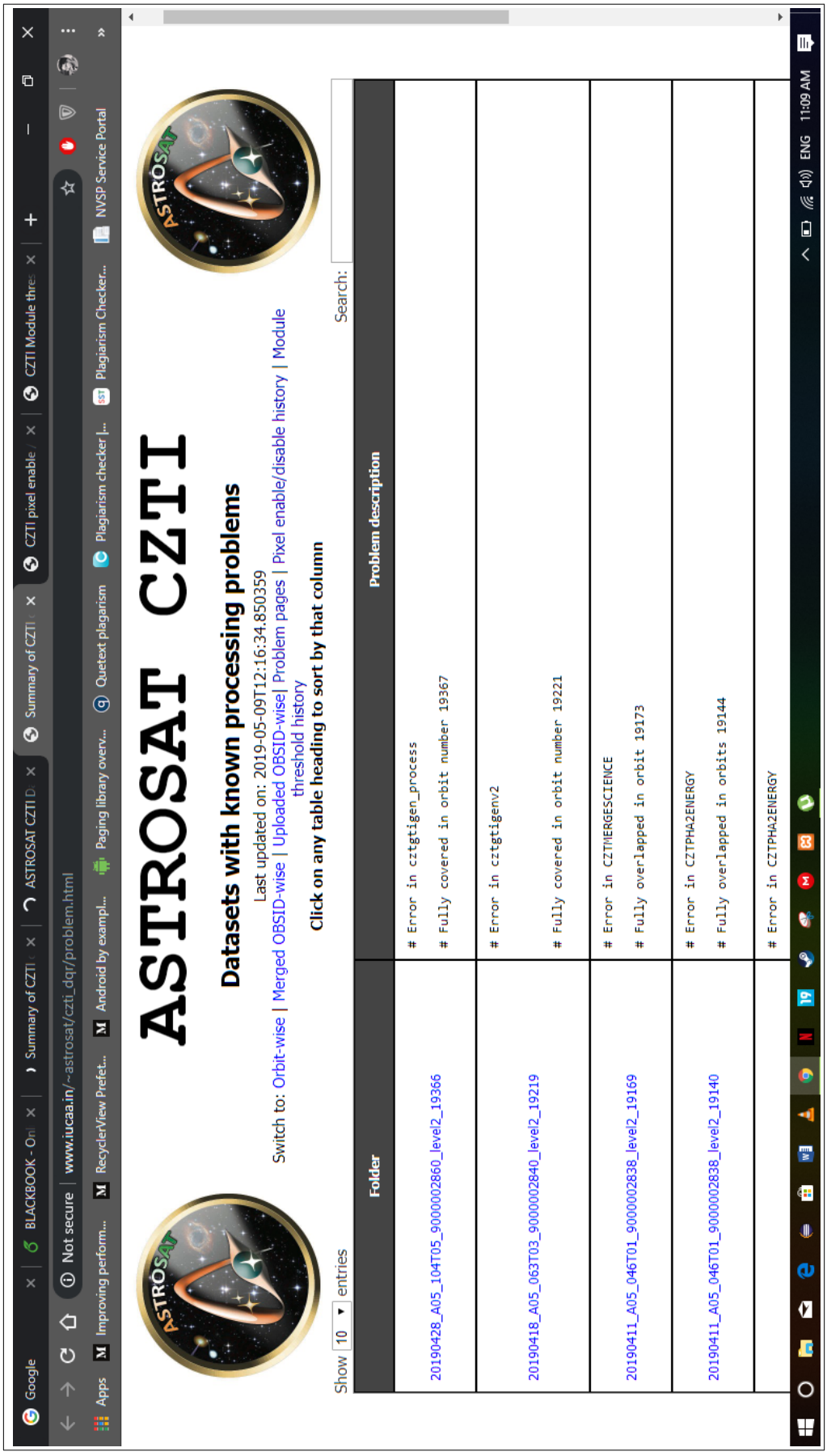
Observation information

- **date-obs**: 2018-12-16
- **time-obs**: 05:02:58.494200000
- **date-end**: 2018-12-16
- **time-end**: 07:28:50.489000000
- **obs_id**: T03_030T01_9000002574
- **exposure**: 3767.75965919
- **sourceid**: AGC 102983
- **observer**: sjanowie
- **ra_pnt**: 8.406667
- **dec_pnt**: 28.73917

^TOP

Basic statistics of files

Param	Original file	Final file
Filename	modeM0/AS1T03_030T01_9000002574_17393cztM0_level2.ev	modeM0/AS1T03_030T01_9000002574_17393cztM0_level2_q

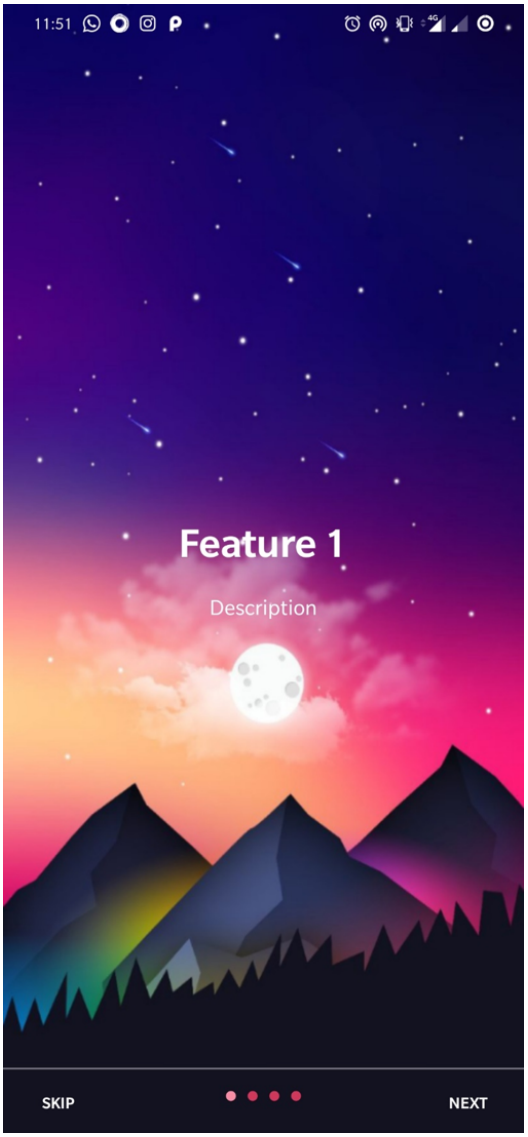

ASTROSAT CZTI
Datasets with known processing problems
Last updated on: 2019-05-09T12:16:34.850359
Switch to: Orbit-wise | Merged OBSID-wise | Uploaded OBSID-wise | Problem pages | Pixel enable/disable history | Module threshold history
Click on any table heading to sort by that column
Search:
Show 10 entries
Folder
Problem description
20190428_A05_104T05_9000002860_level2_19366
Error in cztgtigen_process
Fully covered in orbit number 19367
20190418_A05_063T03_9000002840_level2_19219
Error in cztgtigenv2
Fully covered in orbit number 19221
20190411_A05_046T01_9000002838_level2_19169
Error in CZTMERGESCIENCE
Fully overlapped in orbit 19173
20190411_A05_046T01_9000002838_level2_19140
Error in CZTPHA2ENERGY
Fully overlapped in orbits 19144
Error in CZTPHA2ENERGY

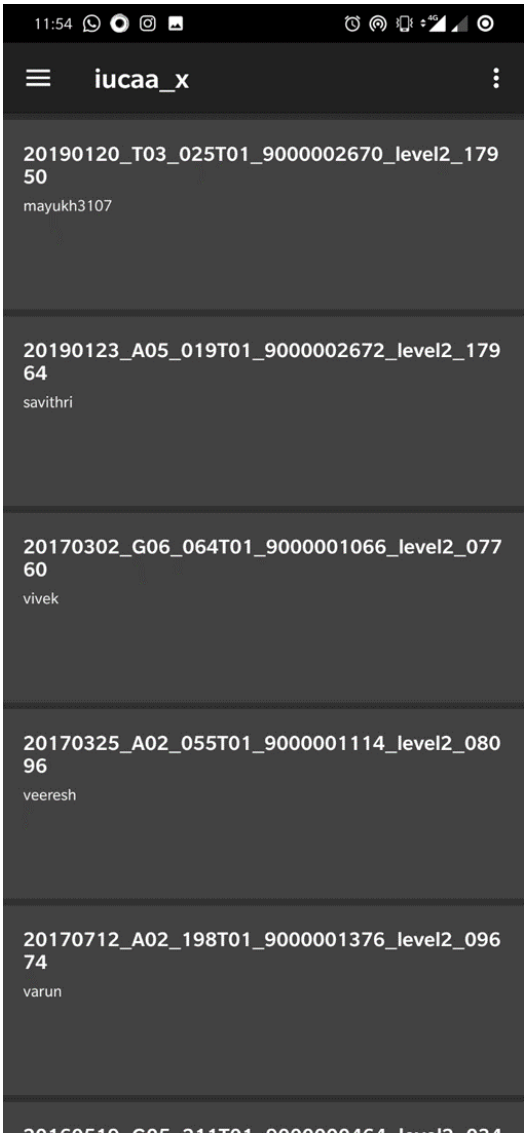
Date (YYYY-MM-DD)	Time (UT) (HH:MM:SS)	Quad (0-3)	Detid (0-15)	Pixid (0-255)	Pixel status (Enable/Disable)
2019-05-01	11:05:46	3	10	66	Disable
2018-08-06	05:25:46	3	13	122	Disable
2018-07-05	05:19:00	3	7	254	Disable
2018-06-21	08:38:00	0	8	15	Disable
2018-03-14	10:38:00	1	4	239	Disable
2018-01-18	04:58:00	0	13	251	Disable
2018-01-17	06:17:00	0	13	151	Disable
2018-01-12	09:37:00	1	2	32	Disable
2018-01-12	09:37:00	1	2	48	Disable
2017-11-27	07:34:00	3	13	152	Disable
2017-11-27	07:34:00	0	8	5	Disable
2017-01-09	11:59:00	2	9	247	Disable
2016-09-26	11:21:17	3	7	237	Disable

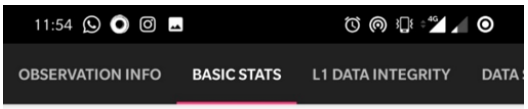
Google
BLACKBOOK - Onl
Summary of CZTI
ASTROSAT CZTI D
Summary of CZTI
CZTI pixel enable /
CZTI Module thres
Not secure | www.iucaa.in/~astrosat/czti_dqr/threshold.html
Apps
Improving perform...
RecyclerView Prefet...
Android by exampl...
Paging library overv...
Quetext plagarism
Plagiarism checker |...
Plagiarism Checker...
NVSP Service Portal
ASTROSAT CZTI
CZTI Module threshold history
This page last updated on: 2016-09-01T15:16:00.338612
Threshold history file SVN revision 1161
Threshold history file date 2016-07-06 16:27:53 +0530 (Wed, 06 Jul 2016)
Switch to: Orbit-wise | Merged OBSID-wise | Problem pages | Pixel enable/disable history | Module threshold history
Click on any table heading to sort by that column
Date (YYYY-MM-DD)
Time (UT) (HH:MM:SS)
Quad (0-3)
Detid (0-15)
Threshold commanded (0-200keV)
Threshold from data (0-200keV)
2016-02-25
06:29:59
0
14
15
12
ENG
11:09 AM

Google
BLACKBOOK -
Summary of C
www.iucaa.in/
Summary of C
CZTI pixel enal
CZTI Module t
dil mein ho tu
Not secure | www.iucaa.in/~astrosat/czti_dqr/20190512_T03_037T01_9000002912_level2_19582/quality.LOG
Apps
Improving perform...
RecyclerView Prefet...
Android by exampl...
Paging library overv...
Quetext plagarism
Plagiarism checker |...
Plagiarism Checker...
NVSP Service Portal
Locating noise-dominated data in quadrant 0
: Time totcounts fraction
Total number of 1 second bins: 7717
Bins with >0 counts : 6473
Bins with >2000 counts : 0
0 of 0 high count rate bins were dominated by noise
Noise dominated 0.00% of total time, 0.00% of detector on time (defined as non-zero counts)
Locating noise-dominated data in quadrant 1
: Time totcounts fraction
: 295406604.0 2555 0.817
: 295411393.0 2079 0.700
Total number of 1 second bins: 7717
Bins with >0 counts : 6473
Bins with >2000 counts : 2
2 of 2 high count rate bins were dominated by noise
Noise dominated 0.03% of total time, 0.03% of detector on time (defined as non-zero counts)
Locating noise-dominated data in quadrant 2
: Time totcounts fraction
: 295406430.0 2343 0.748
Total number of 1 second bins: 7717
Bins with >0 counts : 6472
Bins with >2000 counts : 1
1 of 1 high count rate bins were dominated by noise
Noise dominated 0.01% of total time, 0.02% of detector on time (defined as non-zero counts)
Locating noise-dominated data in quadrant 3
: Time totcounts fraction
: 295406386.0 2110 0.776
: 295406389.0 2369 0.867
: 295406391.0 2444 0.741
: 295406396.0 2773 0.906
: 295406398.0 2051 0.819
: 295406626.0 2520 0.859
: 295406633.0 2295 0.810
: 295406635.0 2637 0.842
: 295406636.0 2819 0.917
: 295406637.0 2378 0.925
ENG
11:14 AM

```
2000
0, 7717, 6473, 0, 0, 0.0, 0.0
1, 7717, 6473, 2, 2, 0.0259168070494, 0.0308975745404
2, 7717, 6472, 1, 1, 0.0129584035247, 0.0154511742892
3, 7717, 6473, 358, 358, 4.63910846184, 5.53066584273
```

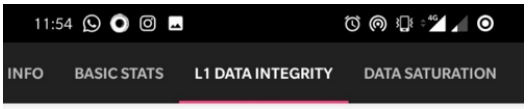






PARAM	ORIGINAL FILE
Filename	modeM0/AS1A05_186T01_9000002692_18133cztM0_level2.
Size (bytes)	503,930,880
Size	480.6 MB
Events in Quadrant A	3,370,702
Events in Quadrant B	3,519,573
Events in Quadrant C	3,398,221
Events in Quadrant D	4,502,832

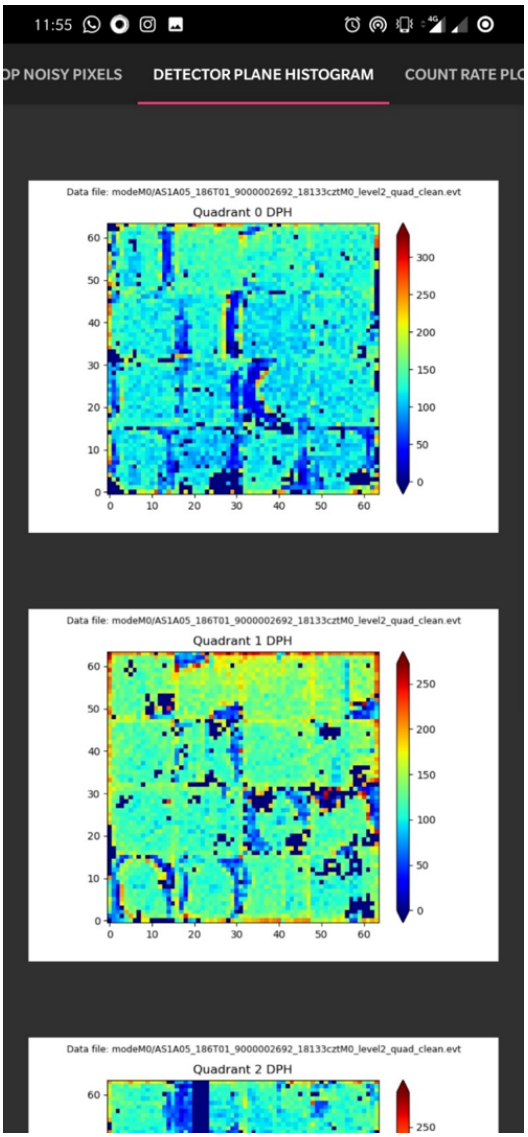
PARAM	FINAL FILE
Filename	modeM0/AS1A05_186T01_9000002692_18133cztM0_level2_
Size (bytes)	76,475,520
Size	72.9 MB
Events in Quadrant A	470,667
Events in Quadrant B	478,358
Events in Quadrant C	450,913
Events in Quadrant D	432,871

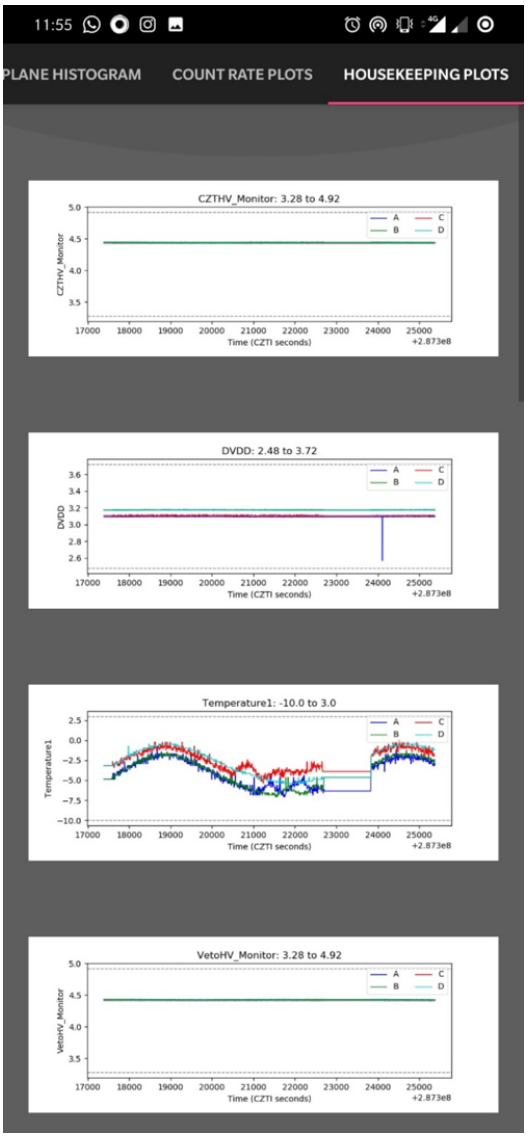


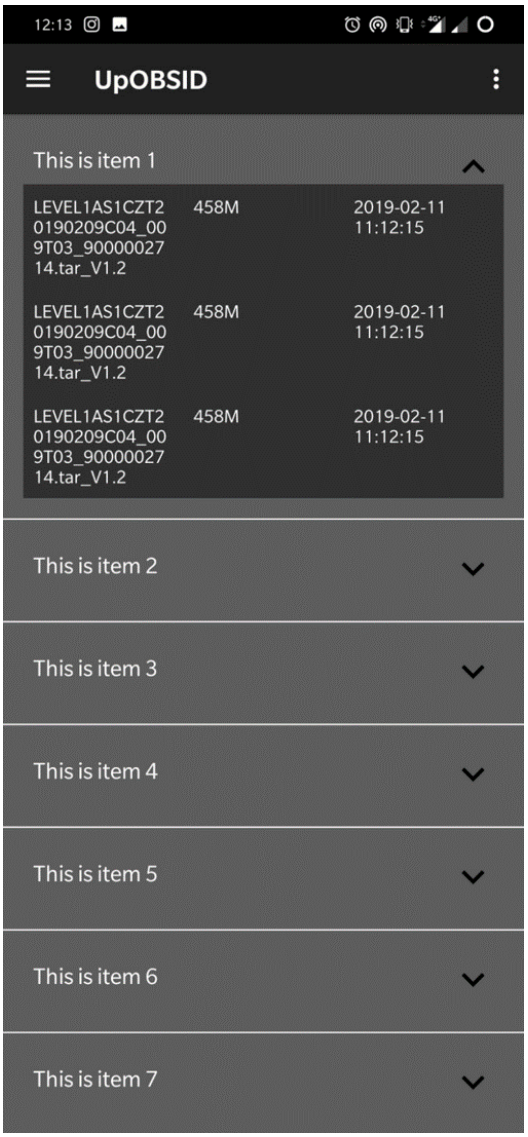
Mode M9			
Quadrant	Bad HDU Flag	Total Packets	Discarded Packets
A	0	188	0
B	0	188	0
C	0	188	0
D	0	188	0

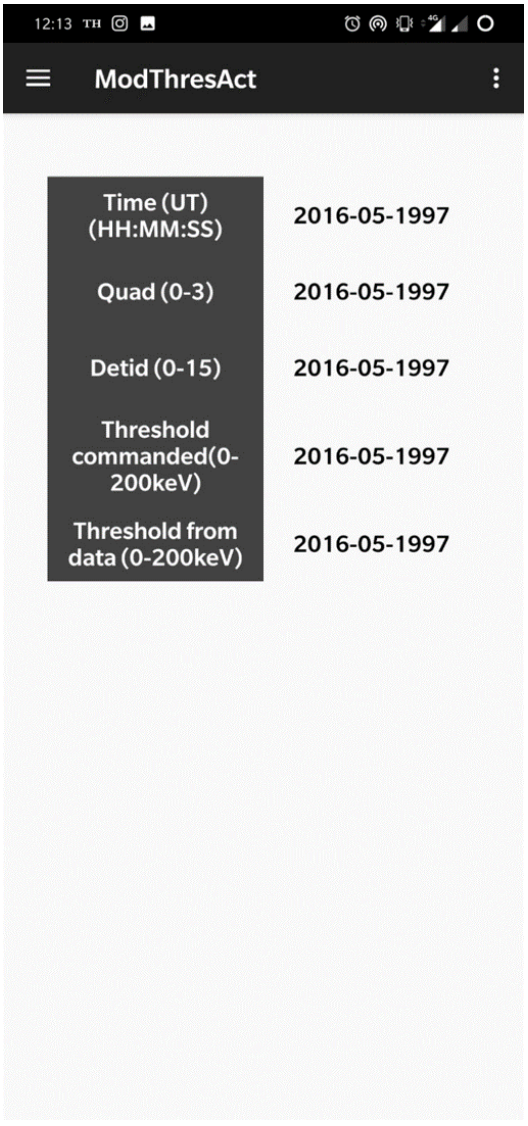
Mode M0			
Quadrant	BAD HDU FLAG	Total Packets	Discarded Packets
A	0	188	0
B	0	188	0
C	0	188	0
D	0	188	0

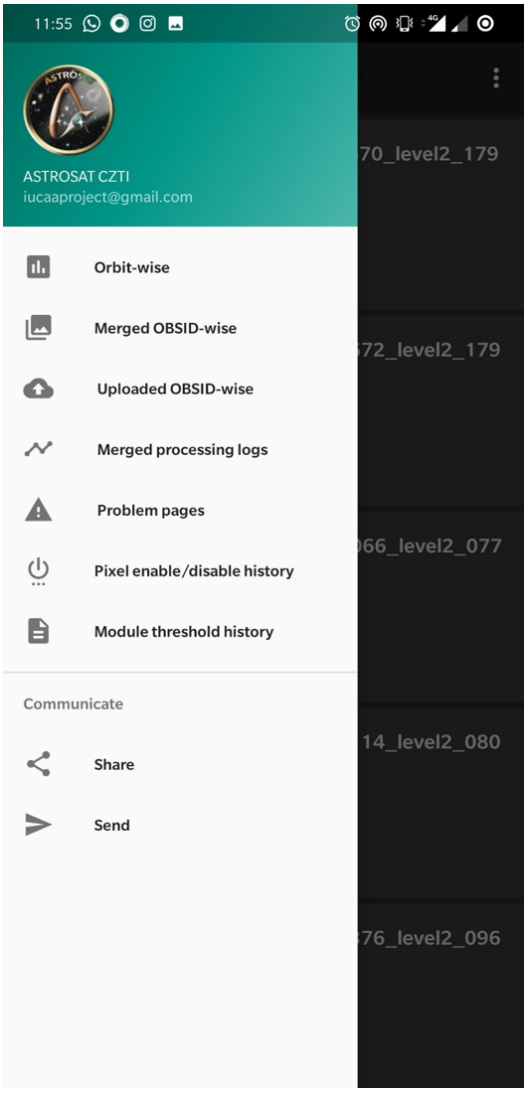
Mode SS			
Quadrant	BAD HDU FLAG	Total Packets	Discarded Packets
A	0	188	0
B	0	188	0
C	0	188	0
D	0	188	0

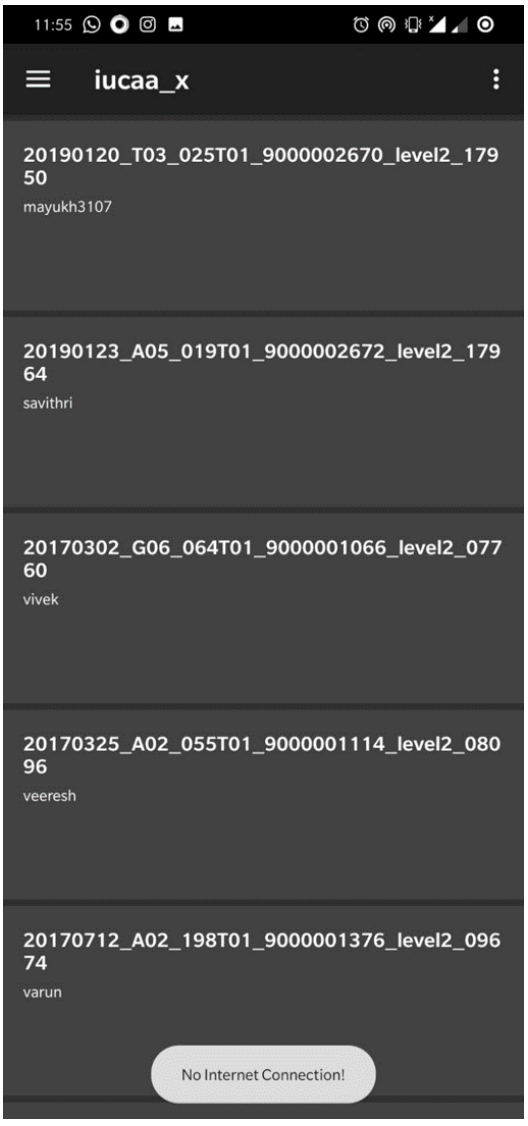

11:55
OP NOISY PIXELS
DETECTOR PLANE HISTOGRAM
COUNT RATE PLO
Data file: modeM0/AS1A05_186T01_9000002692_18133cztM0_level2_quad_clean.evt
Quadrant 0 DPH
Data file: modeM0/AS1A05_186T01_9000002692_18133cztM0_level2_quad_clean.evt
Quadrant 1 DPH
Data file: modeM0/AS1A05_186T01_9000002692_18133cztM0_level2_quad_clean.evt
Quadrant 2 DPH


11:55
PLANE HISTOGRAM
COUNT RATE PLOTS
HOUSEKEEPING PLOTS
CZTHV_Monitor: 3.28 to 4.92
DVDD: 2.48 to 3.72
Temperature1: -10.0 to 3.0
VetoHV_Monitor: 3.28 to 4.92
Time (CZTI seconds)


12:13
UpOBSID
This is item 1
LEVEL1AS1CZT20190209C04_009T03_9000002714.tar_V1.2 458M 2019-02-11 11:12:15
LEVEL1AS1CZT20190209C04_009T03_9000002714.tar_V1.2 458M 2019-02-11 11:12:15
LEVEL1AS1CZT20190209C04_009T03_9000002714.tar_V1.2 458M 2019-02-11 11:12:15
This is item 2
This is item 3
This is item 4
This is item 5
This is item 6
This is item 7


12:13
ModThresAct
Time (UT) (HH:MM:SS) 2016-05-1997
Quad (0-3) 2016-05-1997
Detid (0-15) 2016-05-1997
Threshold commanded(0-200keV) 2016-05-1997
Threshold from data (0-200keV) 2016-05-1997


11:55
ASTROSAT CZTI
iucaaproject@gmail.com
Orbit-wise
Merged OBSID-wise
Uploaded OBSID-wise
Merged processing logs
Problem pages
Pixel enable/disable history
Module threshold history
Communicate
Share
Send
70_level2_179
72_level2_179
66_level2_077
14_level2_080
76_level2_096


11:55
iucaa_x
20190120_T03_025T01_9000002670_level2_17950
mayukh3107
20190123_A05_019T01_9000002672_level2_17964
savithri
20170302_G06_064T01_9000001066_level2_07760
vivek
20170325_A02_055T01_9000001114_level2_08096
veeresh
20170712_A02_198T01_9000001376_level2_09674
varun
No Internet Connection!

Chapter 5

MATHEMATICAL MODEL

5.1 Linear Programming Model

The proposed mathematical model is based on linear programming paradigm which is used for performance optimization and better time utilization. The proposed paradigm works on the basic statistical principle of finding the maximum values. The paradigm can help in understanding the resources in environments such as cloud and utilizing them at optimal level.

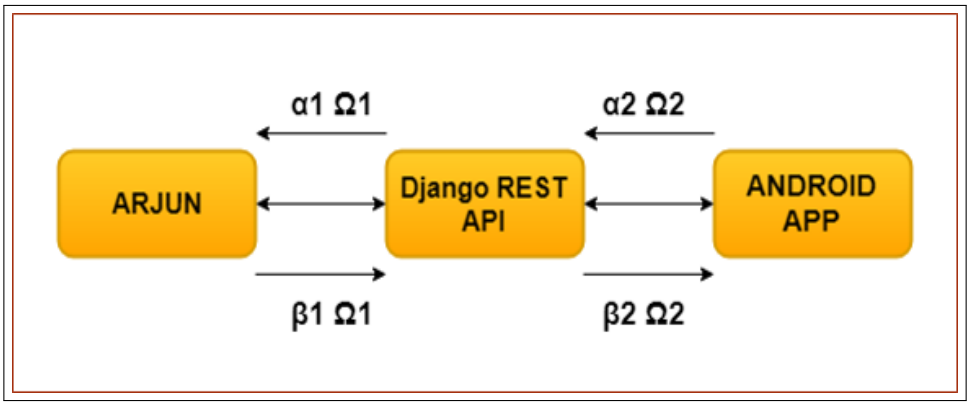


Figure 5.1: Linear Programming Representation

The notations used for the proposed model are as follows; notations can be used to equate the linear programming model which helps in solving maximum resources and minimum time utilization problem.

$$\mathbf{minimize(z) = \lambda 1\ \Omega 1 + \lambda 2\ \Omega 2} \tag{5.1}$$

Table 5.1: Linear Programming Model

Resources	**Data Center**	**Web Service**	**Existing Time**
Resource 1(α)	$\alpha 1$ $\Omega 1$	$\alpha 2$ $\Omega 2$	$\lambda 1$
Resource 2(β)	$\beta 1$ $\Omega 1$	$\beta 2$ $\Omega 2$	$\lambda 2$
Minimize(Ω)	$\Omega 1$	$\Omega 2$	

$$\boldsymbol{\alpha 1\ \Omega 1 + \alpha 2\ \Omega 2 \leq \lambda 1} \tag{5.2}$$

$$\boldsymbol{\beta 1\ \Omega 1 + \beta 2\ \Omega 2 \leq \lambda 2} \tag{5.3}$$

where α = Total running time of resource type one

β = Total time running of resource type two

λ = Total Time Resources Can be processed

Ω = Maximum Utilization of Resources

Chapter 6

SYSTEM DIAGRAMS

6.1 UML Diagrams

6.1.1 Use Case Diagram

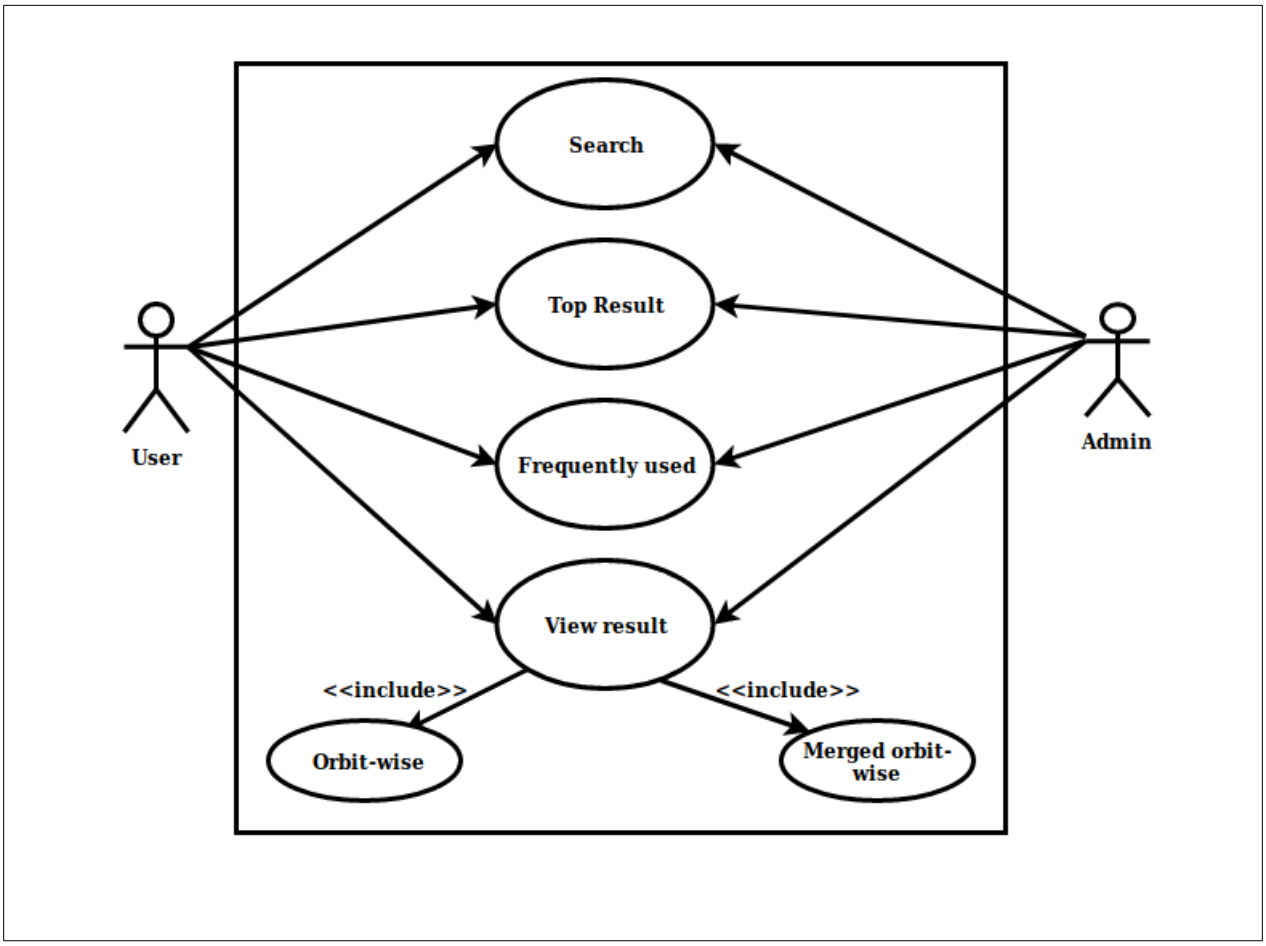


Figure 6.1: Linear Programming Representation

Use case diagrams are usually referred to as behavior diagrams used to describe a

set of actions (use cases) that some system or systems (subject) should or can perform in collaboration with one or more external users of the system (actors).

The various tasks performed by actors in our system are described in the diagram above.

6.1.2 Sequence Diagram

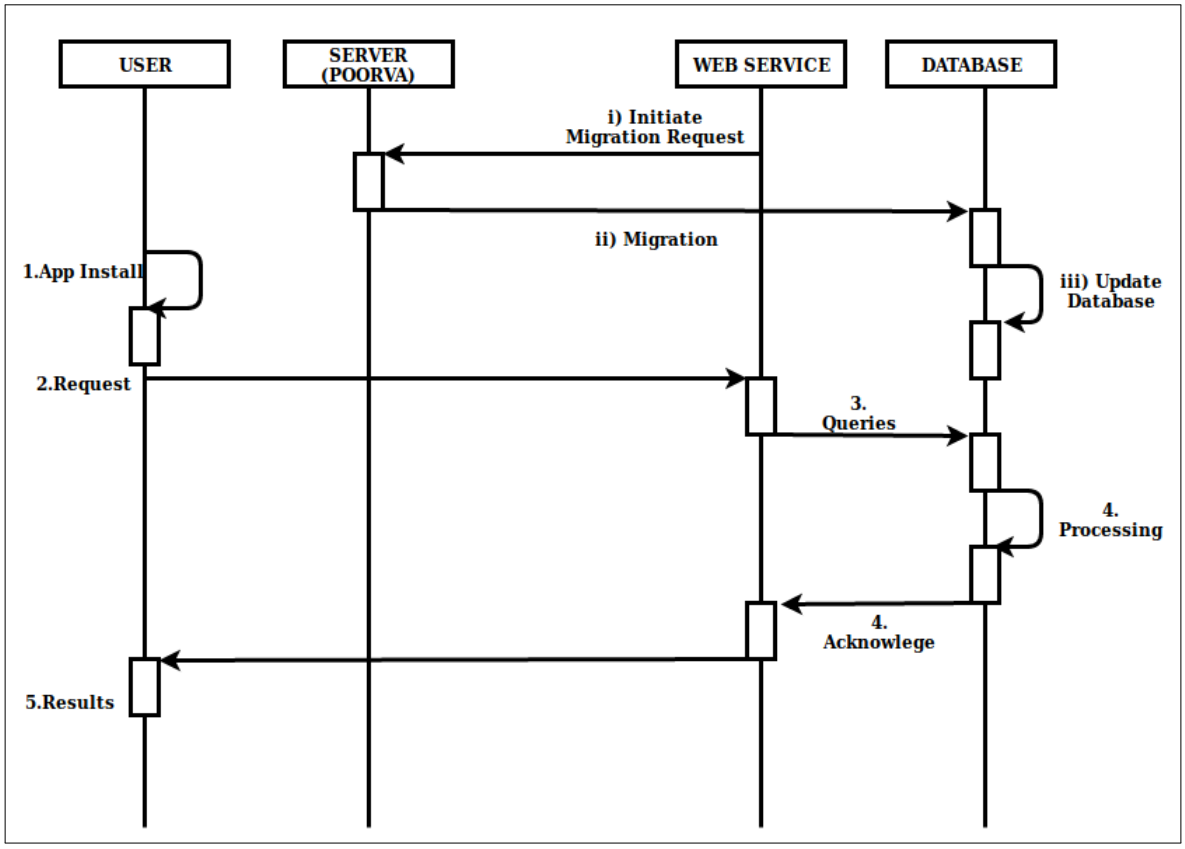


Figure 6.2: Linear Programming Representation

Sequence diagrams are sometimes called event diagrams or event scenarios.A sequence diagram shows, as parallel vertical lines (lifelines), different processes or objects that live simultaneously, and, as horizontal arrows,the messages exchanged between them, in the order in which they occur.

The diagram below shows the exchange of messages between various entities in the system.

6.1.3 ER Diagrams

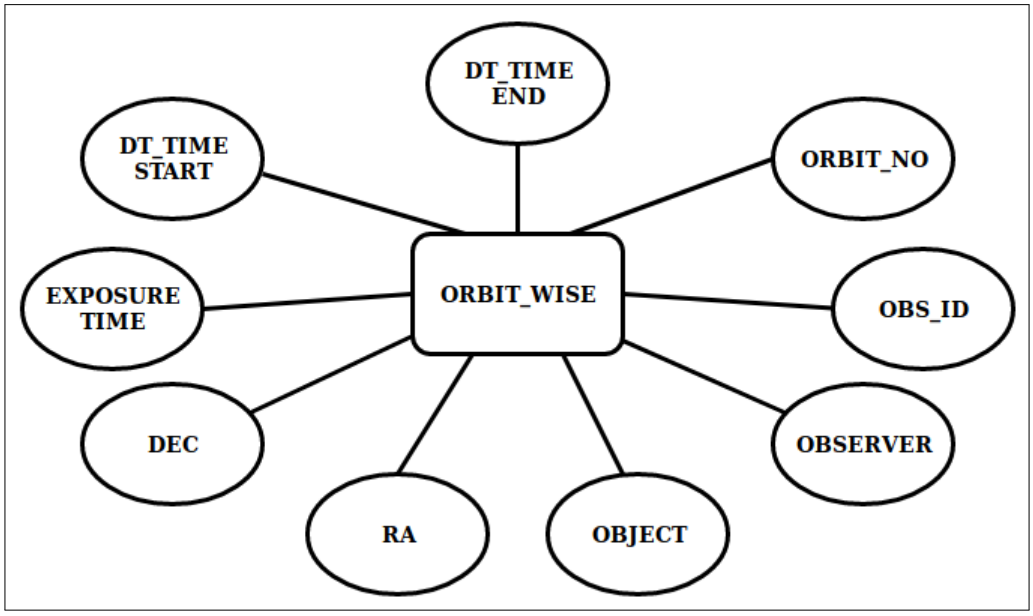


Figure 6.3: Entity Relationship

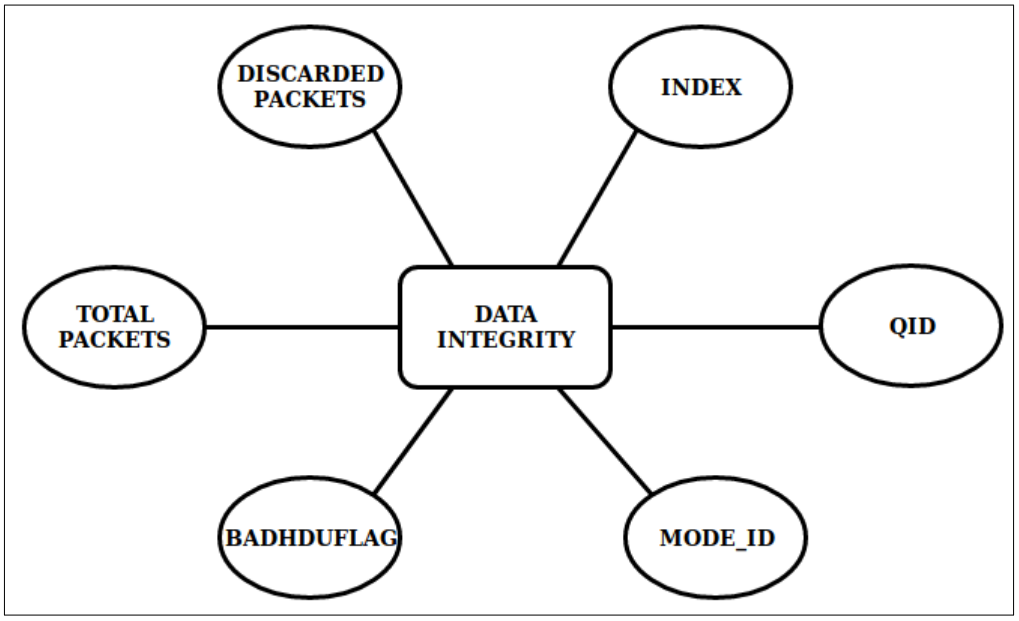


Figure 6.4: Data Integrity Diagram

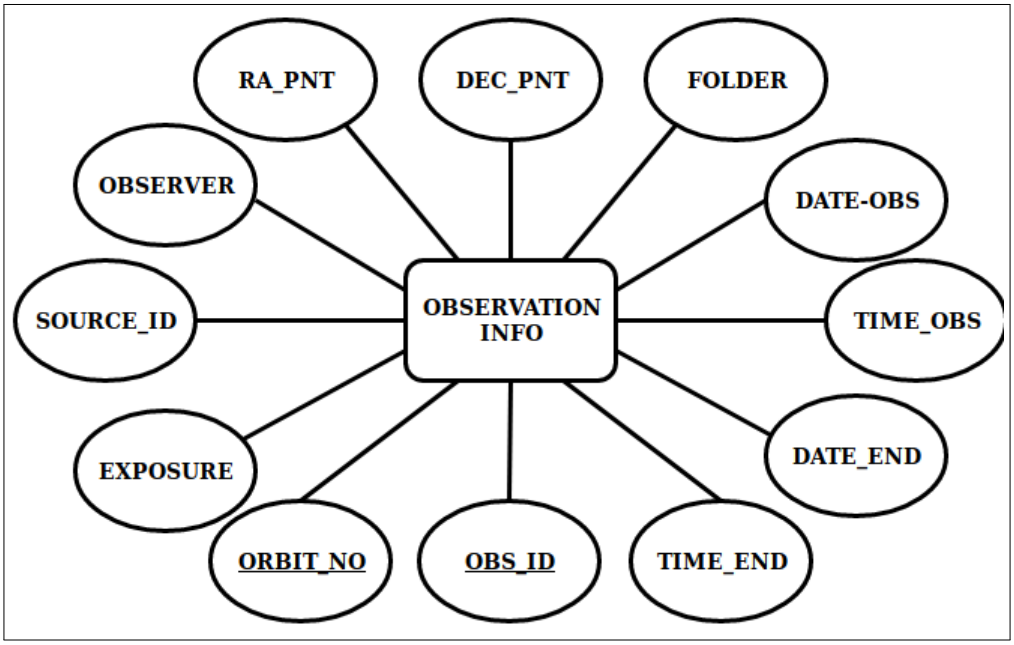


Figure 6.5: Observation Info

6.2 Data Flow Diagram

A data flow diagram (DFD) breaks down the main processes into subprocesses that can then be analyzed and improved on a more intimate level. It can be used to plan or record the specific makeup of a system. The diagram below shows a detailed flow of data through the entire system.

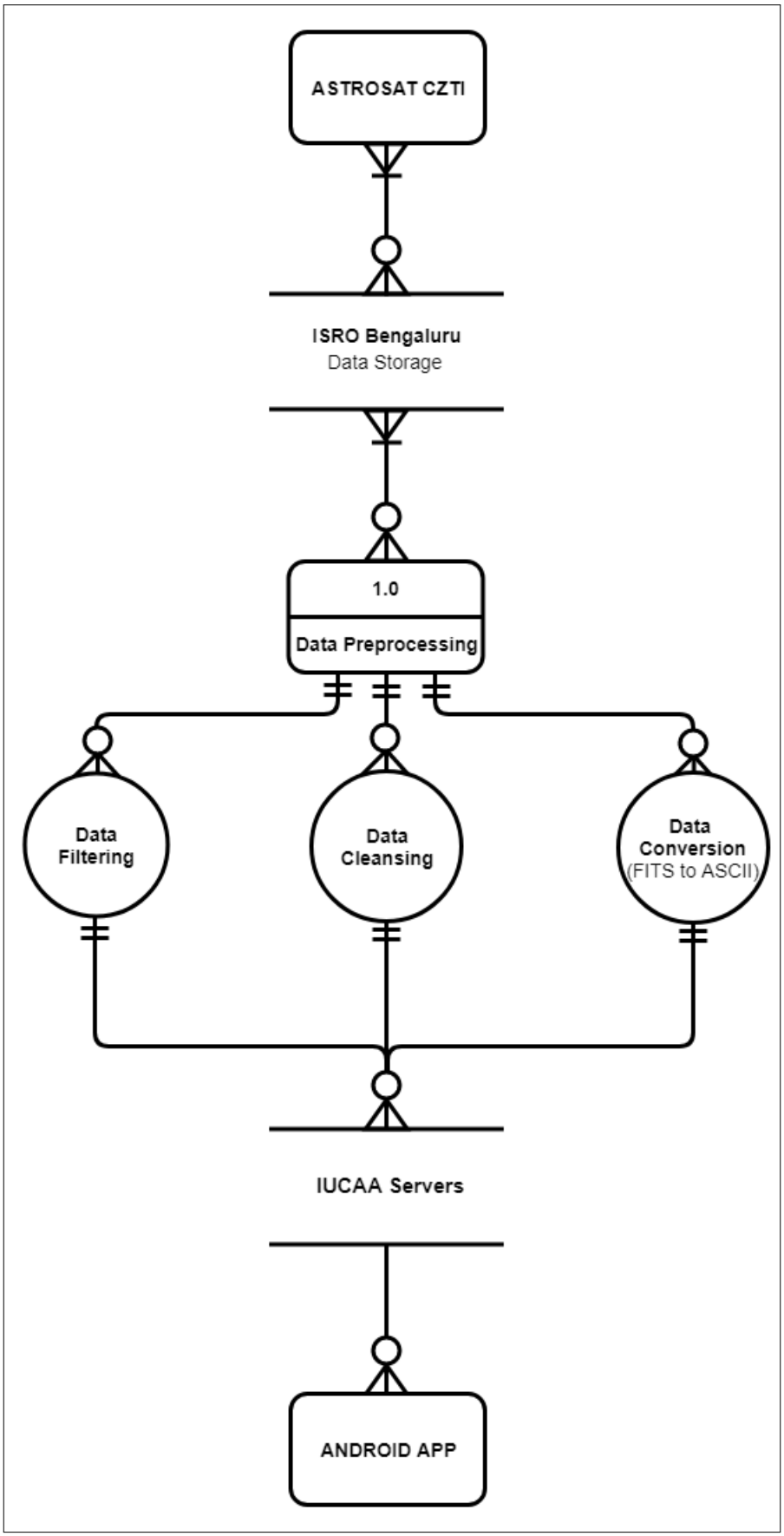


Figure 6.6: Data Flow Diagram

Chapter 7

TESTING

Test Beds Used :

- **AWS Device Farm :**

  AWS Device Farm is an app testing service that allows to test and interact with the Android, iOS, and web apps on many devices at once, or reproduce issues on a device in real time. It allows the user to View video, screenshots, logs, and performance data to pinpoint and fix issues and increase quality before shipping the app. AWS Device Farm lets the user test the application on a shared fleet of 2500+ devices or on the private device lab in the cloud.

- **Android Virtual Device**

  It is an inbuilt functionality within Android Studio.

- **Functional Testing**

  Testing is normally achieved by user interface initiated test flows. Not just the flow of a use case is tested, but the various business rules are also tested. Testing is done by certifying the requirements. i.e. whether the application is working based on the requirements.

Test Case 1: Location based testing using AWS Device Farm

The android application was virtually tested by entering co-ordinates of different Cities in AWS Device Farm.

Test Case 2: Device compatible testing using AWS Device Farm

The application developed was checked for variable device compatibility.

Test Case 3: Performance testing based on network connectivity

During this testing, request/response to/from the service was tested for various conditions. It was mainly done to verify the response time in which the activity is performed like refreshing data after sync etc. This was done for both strong wifi connection and the mobile data network & is an in-house testing. The app was tested with both high (2 Mbps) and low (128 kbps) speed internet.

Test Case 4: Interrupt Testing

This type of testing is also known as Offline Scenario Verification wherein the communication breaks in the middle. Verified that the app does behave as designed/desired if the device is connected to the internet through Wi-Fi. Verified if the application communicates with a webserver, if the network coverage fails, does the application automatically reinitiate the communication once the coverage is re-established or not. Verified Stability check i.e. if the app has a list (e.g. pictures) in it, try scrolling through it at high speed.

Test Case 5: Alpha testing

The android application was tested thoroughly by the development team in various scenarios in-house.

Test Case 6: Beta testing

Beta testing is performed by a third party who was not involved in the development of the android application. In this case, the beta testing was performed by the peers of the development team.

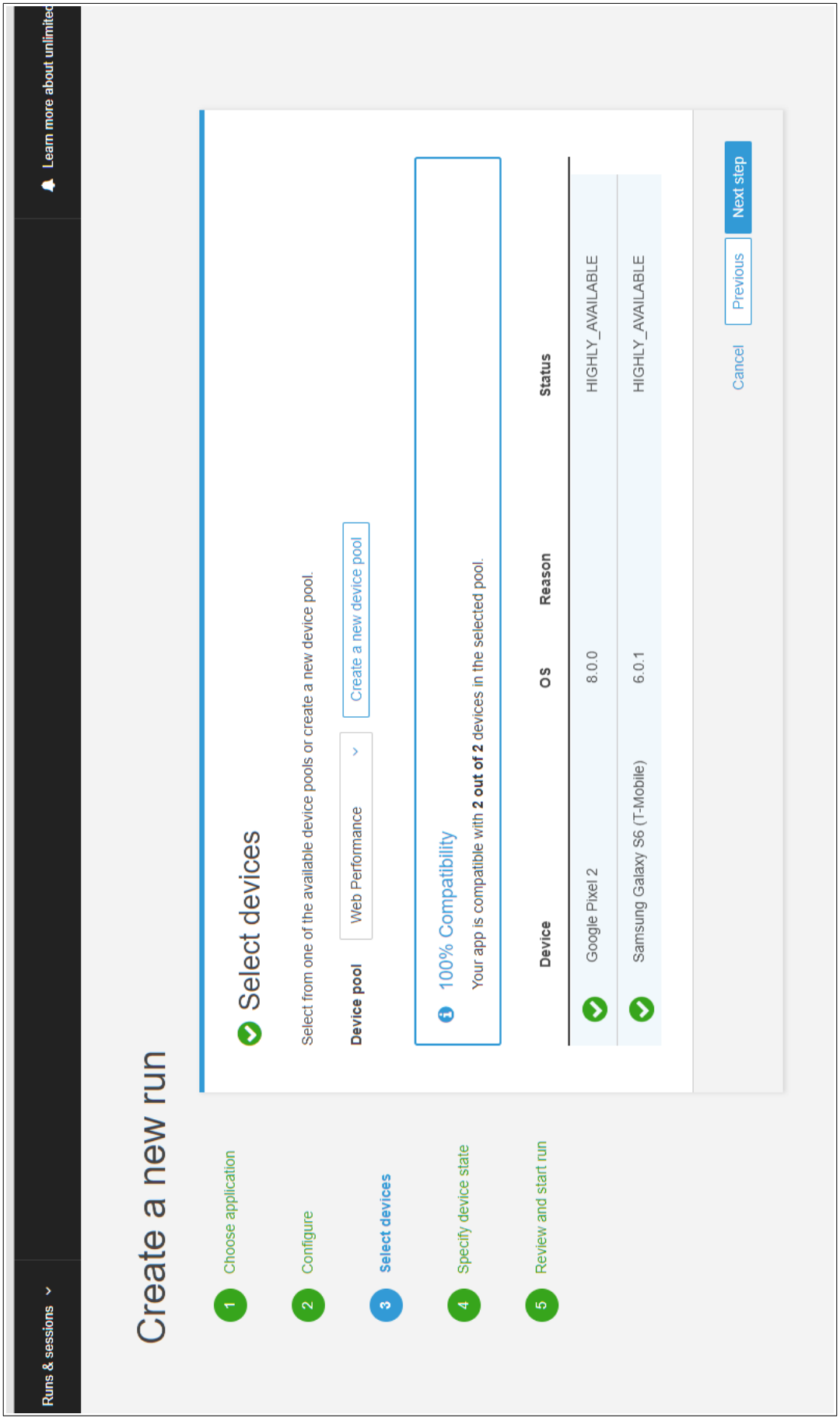


Figure 7.1: Web Performance Testing

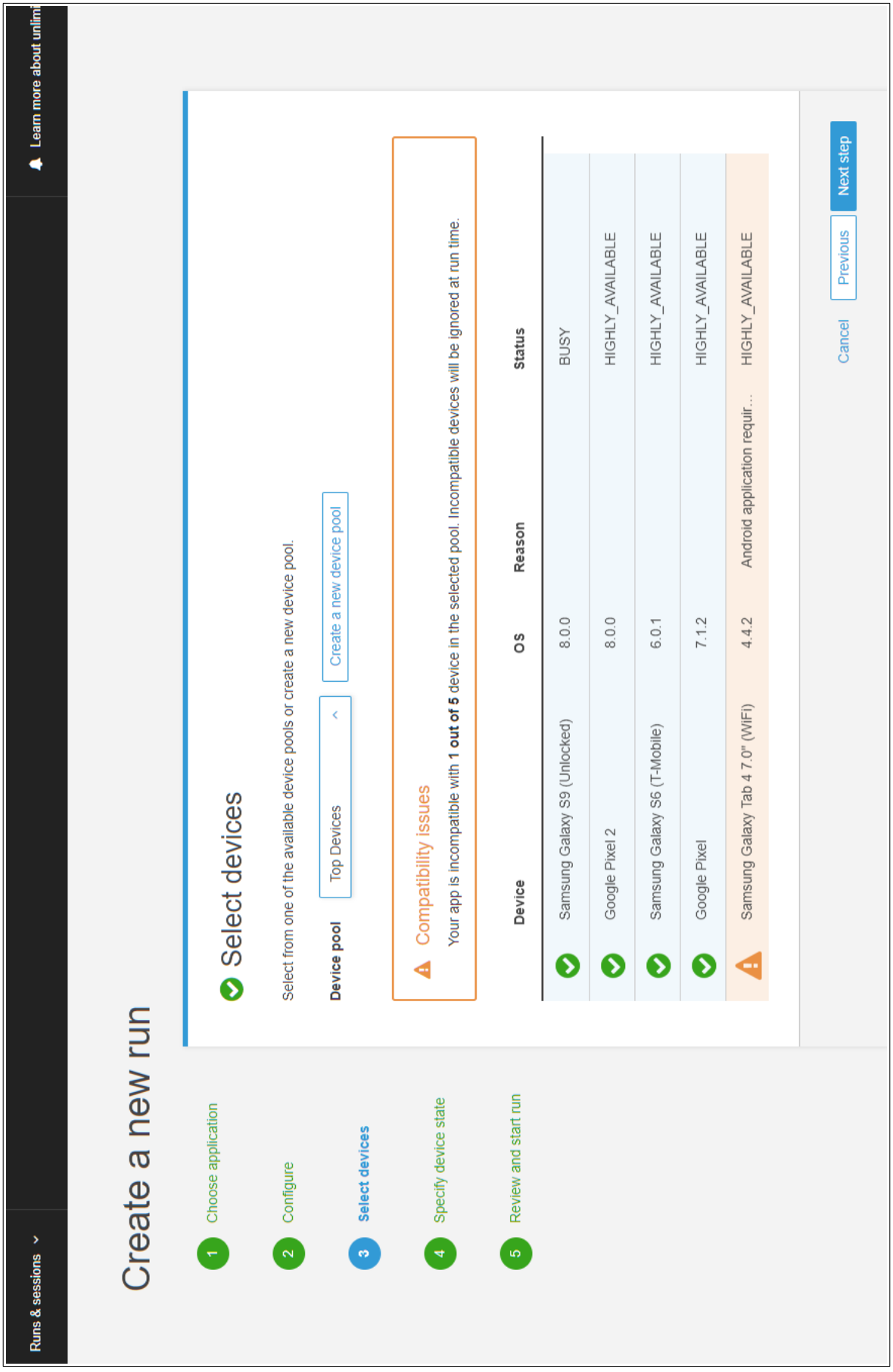


Figure 7.2: Device Compatible testing

Chapter 8

COMPARATIVE ANALYSIS

The following table and graph clearly demonstrates the improvement achieved through our proposed system over the existing architecture. The graph of the existing architecture shows exponential increase whereas the proposed architecture gives us a linear minuscule increase. For any data driven system, the number of records to be fetched is directly proportional to the time required to fetch them. The proposed system tries to minimize the scaling time factor of the directly proportional relation. The graph of the existing system after some span of records will result in unrealistic delay. This evidently proves that the employed amalgamation of technologies like Django, REST Framework and ARJUN application server along with the efficient use of Volley web services enables a very fruitful approach towards architectural optimization.

$$Slope = \frac{y2 - y1}{x2 - x1} \tag{8.1}$$

$$Slope\ of\ existing\ architecture = \frac{170.3 - 120.93}{35000 - 25000} \tag{8.2}$$

$$Slope\ of\ existing\ architecture = 0.004937 \tag{8.3}$$

$$Slope\ of\ proposed\ architecture = \frac{4.49 - 3.38}{35000 - 25000} \tag{8.4}$$

$$Slope\ of\ proposed\ architecture = 0.000111 \tag{8.5}$$

Table 7.1: Preliminary analysis

Records	Time(existing) in secs	Time(proposed) in secs
0	0	0
5000	26.18	1
10000	52.37	2
15000	78.55	2.73
20000	100.74	3.33
25000	120.93	3.38
30000	142.11	4.17
35000	170.30	4.49
40000	209.48	4.82
45000	248.67	5.01
50000	310.86	5.49

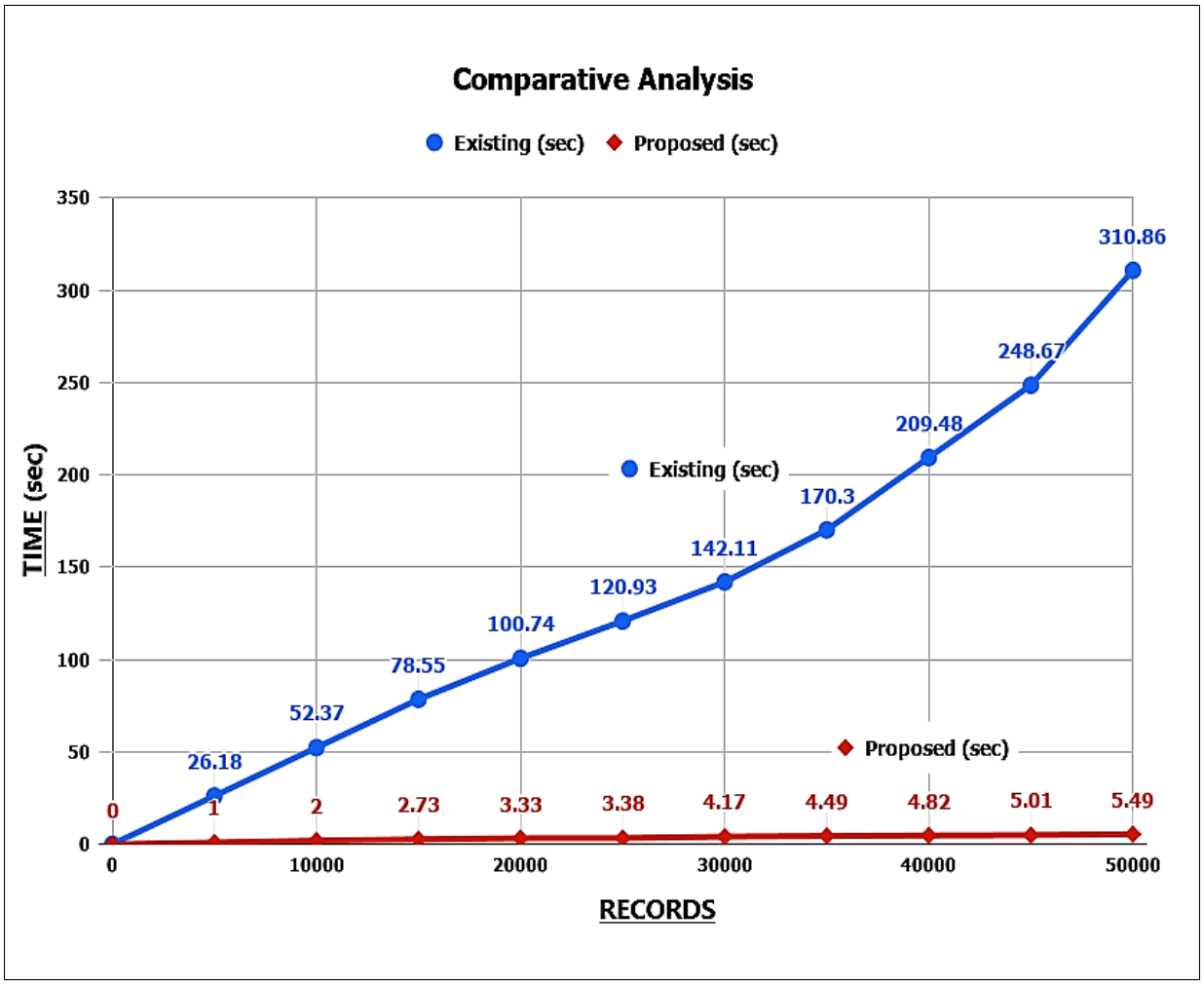


Figure 8.1: Comparative Analysis of Existing System and Proposed System

Chapter 9

ADVANTAGES & LIMITATIONS

9.1 Advantages

- **Reduced Time Delay:**

  A RDBMS is a software system that provides access to a relational database. The software system is a collection of software applications that can be used to create, maintain, manage and use the database. A "relational database" is a database structured on the "relational" model. Data are stored and presented in a tabular format, organized in rows and columns with one record per row. Optimizations built into an RDBMS, and the design of the databases, enhance performance, allowing RDBMSs to perform more than fast enough for most applications and data sets. Improvements in technology, increasing processor speeds and decreasing memory and storage costs allow systems administrators to build incredibly fast systems that can overcome any database performance shortcomings.

- **Consistent Data Structure:**

  The table format is simple and easy for database users to understand and use. RDBMSs provide data access using a natural structure and organization of the data. Database queries can search any column for matching entries.

- **Multi-User Access:**

  RDBMSs allow multiple database users to access a database simultaneously. Built-in locking and transactions management functionality allow users to access data as it is being changed, prevents collisions between two users updating the data, and keeps users from accessing partially updated records.

- **Network Access:**

  RDBMSs provide access to the database through a server daemon, a specialized software program that listens for requests on a network, and allows database clients to connect to and use the database. Users do not need to be able to log in to the physical computer system to use the database, providing convenience for the users and a layer of security for the database. Network access allows developers to build desktop tools and Web applications to interact with databases.

- **Maintenance:**

  RDBMSs feature maintenance utilities that provide database administrators with tools to easily maintain, test, repair and backup the databases housed in the system. Many of the functions can be automated using built-in automation in the RDBMS, or automation tools available on the operating system.

- **Increased Portability:**

  The newly developed system architecture for the storage of ASTROSAT DQRs will make the data portable, as in, movement of the data from one system to another will be possible without much fuss and data could be accessed from remote devices.

- **Increased Scalability:**

  The amount of data that can be stored will be very flexible due to the use

of RDBMS for the storage of the same. The system will respond with equal efficiency to varied amount of data sizes.

- **Ease of Use and Navigation:**

  The application developed will be easy and intuitive to use and navigate through which will enable the users to access the data with great ease and speed compared to the older system.

- **Public Access:**

  The ASTROSAT data will be available for public access so that a varied class of users from scientists to amateur researchers would be able to use the data for constructive purposes due to the availability of the data on the android and web platform. The data will be updated at real time so that the latest orbital data would be available as early as possible to the users. Reduced time delay will further aid in integrating the data with other applications wherever required. (Built in security can be introduced in the android application through personalized login for enhancing data security but, IUCAA authorities do not coerce the implementation of the same.)

- **Low-cost EDI (Electronic Data Interchange):**

  Electronic Data Interchange (EDI) is the electronic interchange of business information using a standardized format; a process which allows one company to send information to another company electronically rather than with paper. Business entities conducting business electronically are called trading partners.

- **Remote status reports through the Internet:**

  The status of the server, web service, database and android application could be monitored remotely due to the implementation of web based architecture.

- **Code Reusability:**

  This system can be implemented for similar problem definitions requiring management of huge data and rapid access of the data through a portable medium.

- **Interoperability:**

  Web services allow various applications to talk to each other and share data and services among themselves. Other applications can also use the web services. For example, a VB or .NET application can talk to Java web services and vice versa. Web services are used to make the application platform and technology independent.

- **Standardized Protocol:**

  Web services use standardized industry standard protocol for the communication. All the four layers (Service Transport, XML Messaging, Service Description, and Service Discovery layers) use well-defined protocols in the web services protocol stack. This standardization of protocol stack gives the business many advantages such as a wide range of choices, reduction in the cost due to competition, and increase in the quality.

- **Low Cost Communication:**

  Web services use SOAP over HTTP protocol, so you can use your existing low-cost internet for implementing web services. This solution is much less costly compared to proprietary solutions like EDI/B2B. Besides SOAP over HTTP, web services can also be implemented on other reliable transport mechanisms like FTP.

9.2 Limitations

- **Requires stable internet connection:**

  The android application is web based and requires a stable internet connection of at least 3rd Generation to perform efficiently without which requests for data retrieval would not reach the web service and displaying the ASTROSAT data would become difficult.

- **Minimum android version 5.0 required:**

  The mobile application requires minimum Android version 5.0 as all the required functionalities cannot be implemented using a lower android sdk. The user must have a phone having android version greater than 5.0 (Android-Lollipop) to use the app and access the DQRs.

Chapter 10

CONCLUSION & FUTURE SCOPE

In this project, an architecture for the ASTROSAT-CZTI DQRs employing a web service based on Django web framework and RESTful API; Application Server consisting of a relational database; A client-side user interface comprising of a mobile interface and desktop interface is propositioned that optimizes the storage and fetching mechanism of the DQRs. It enhances the speed of the system significantly. The load balancing and dynamic data fetching process employed in the Django and RESTful API allows the mobile application to maintain a lightweight architecture. It also makes the data considerably ubiquitous compared to the existing system owing to the mobile user interface it provides to its users. These implementations have resulted in establishment of a highly scalable and robust system architecture.

Chapter 11

APPENDIX A : Satisfiablity Analysis

Informally an algorithm is any well-defined computational procedure that takes some value or a set of slues as input and produces some value or a set of values as output. How- ever, finding the optimized solution to a problem is of utmost important in the field of computer science. Approximations and heuristics are more than welcome in applications, since they oftentimes provide efficient solutions for special cases. Theoretically, however, the computer science community is still lacking a good understanding of the computational difficulties of this problem though many facts about it are already known.Time complexity is simply a measure of the time it takes for a function or expression to complete its task, as well as the name of the process to measure that time. In simple words, every piece of code we write, takes time to execute. The time taken by any piece of code to run is known as the time complexity of that code. The lesser the time complexity, the faster the execution. Hence, time complexity allows us to compare the efficiency of algorithms. Time Complexity: P, NP, NP-Hard, NP-Complete

11.0.1 P Class Problem

Problems in class P can be solved with algorithms that run in polynomial time. If the running time is some polynomial function of the size of the input, for instance if the algorithm runs in linear time or quadratic time or cubic time, then we say the algorithm runs in polynomial time and the problem it solves is in class P. You can also have algorithms that run in quadratic time O(n ^{2}), exponential time O(2 ^{n}), or even logarithmic time O(log n). Binary search (on a balanced tree) runs in logarithmic time because the height of the binary search tree is a logarithmic function of the number of elements in the tree.

11.0.2 NP Class Problem

Programs that dont (necessarily) run in polynomial time on a regular computer, but do run in polynomial time on a nondeterministic Turing machine. These programs solve problems in NP, which stands for nondeterministic polynomial time. A nondeterministic Turing machine can do everything a regular computer can and more. This means all problems in P are also in NP. An example in which checking the answer is faster than finding the answer would be finding the solution to a multivariable system of equations. In this case, one need only substitute the solution values in for the variables in the equations and check that all of the equations are indeed satisfied. Typically, solving such a system is very difficult, if it is even exactly solvable at all.

NP Hard Problem

If a problem is NP-hard, this means we can reduce any problem in NP to that problem. This means if we can solve that problem, we can easily solve any problem in NP. If we could solve an NP-hard problem in polynomial time, this would prove P = NP. An eminently practical example is the Traveling Salesman Problem: Given a bunch

of cities on a map and all the roads that connect them, find the shortest route that visits all of the cities at least once. Note that this problem, as stated, is not in NP. There is no efficient way to prove that a proposed solution is indeed the shortest. However, there is a way to take any problem in NP and create a map for which the shortest route through the cities is equivalent to a solution to the original problem.

NP Complete Problem

NP-complete problems are in NP, the set of all decision problems whose solutions can be verified in polynomial time; NP may be equivalently defined as the set of decision problems that can be solved in polynomial time on a non-deterministic Turing machine. A problem p in NP is NP-complete if every other problem in NP can be transformed (or reduced) into p in polynomial time.

This identified problem of reducing the delay time of the responses was solved in polynomial time with the time complexity of O(n). Hence the objective of the problem falls under the NP-Complete category of problems.

Chapter 12

APPENDIX B : Research and Awards

12.1 Journal : IEEE

The following paper has been accepted for the IEEE conference held in IIT-Kanpur(July,2019).

Paper Title

Architectural Optimization of Large Scale Astronomical Data

Conference

THE 10th INTERNATIONAL CONFERENCE ON COMPUTING, COMMUNICATION AND NETWORKING TECHNOLOGIES (ICCCNT) 2019, July 6-9, 2019 at Indian Institute of Technology, Kanpur (IIT-K).

Authors

Brijesh Choudhary, Chinmay Pophale, Ankit Dani, Aditya Gutte, Vandana Jagtap

Abstract

Satellites all around the globe generate huge quantities of data periodically on a daily basis. The management of such large quantities of data poses significant problems ranging from data storage, retrieval and manipulation. The current scenario of managing Indian Space Research Organisation's (ISRO) ASTROSAT-CZTI satellite data involves an elementary and unsophisticated approach which causes a lot of delay in

data retrieval and further causes noticeable data portability issues. The evaluation and research on the above mentioned complication paved a way for us to introduce a new architectural system to solve this complex optimization problem. We propose the creation of an independent application server and an underlying Django REST API based web service which would facilitate the efficient management and retrieval of this data. This also extends the scope, making the system more dynamic and helps in load balancing. The application server helps the system to regulate the turbulent flow of huge data quantities. The Django web framework provides feature distribution which helps manage unorganised data cost effectively. Thus, it removes the possible causes of deterioration of response in the existing system. The REST API extends the scope of the system by providing umpteen useful features which makes universal interfacing of our system possible. The techniques used in the development of our proposed system including data migration, prefetching of data and minimizing the time trade-off, address all existing challenges and show a significant improvement in the overall performance of the system.

12.2 Journal : Springer

The following paper is in the process of being evaluated.

Paper Title

Comparative Analysis for an Optimized Data Driven System

Conference

International Conference on Computational Science and Applications(ICCSA 2019), held at Dr. Vishwanath Karad MIT World Peace University (MIT-WPU), Pune, INDIA. August 7-9, 2019.

Authors

Chinmay Pophale, Brijesh Choudhary, Ankit Dani, Aditya Gutte, Vandana Jagtap

Abstract

While designing huge systems based on data access, storage and retrieval, there a lot of problems faced by developers including ensuring portability, cross platform support, scalable design, secure platforms and reduced latency and redundancy. These problems and need to discussed and tackled in order to increase performance and efficiency for the benefit of the end user. This study makes use of one such data driven problem involving huge quantities of astronomical data to demonstrate the advantages of using umpteen efficient technologies and services in order to achieve significant improvement in results. Further sections compare and analyse the pros and cons of different technologies and protocols in use in order to derive fruitful conclusions.

12.3 Best Final Year Project

Competition Details

MIT-WPU Texephyr at MIT-WPU(15th and 16th March,2019)

Project Name

Architectural Optimization of ASTROSAT CZTI.

12.4 Participation

Competition Details

Computer Society of India Regional Level Project Competition 2019 in collaboration with ICERTIS and Soft-Corner,Pune held at D. Y. Patil Institute of Technology

Project Name

Architectural Optimization of ASTROSAT CZTI.

Chapter 13

APPENDIX C : Plagiarism Report

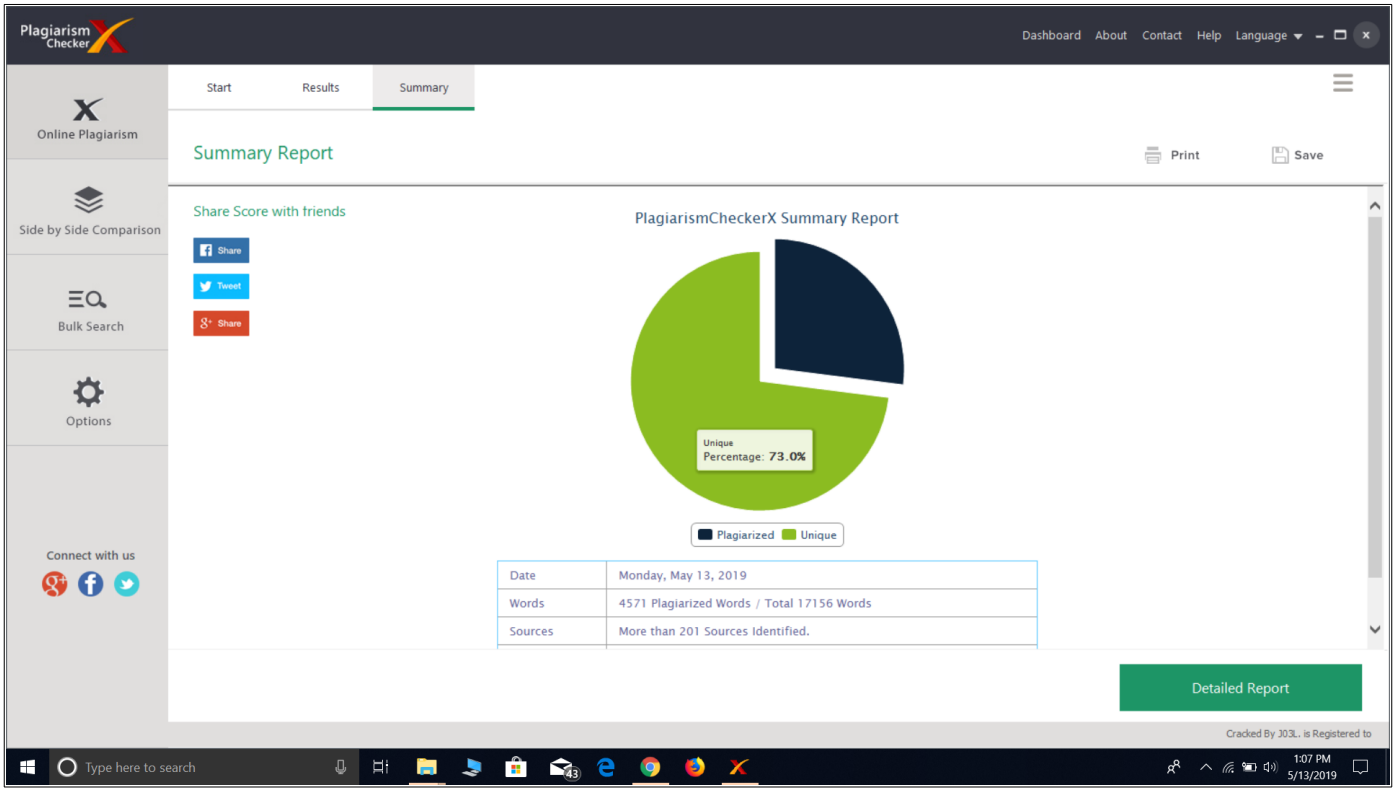


Figure 13.1: Plagiarism Report

Bibliography

[1] J. C. French and A. L. Powell, "Metrics for evaluating database selection techniques" Proceedings. Tenth International Workshop on Database and Expert Systems Applications. DEXA 99, Florence, Italy, 1999, pp. 726-730.

[2] R. Valarezo and T. Guarda, "Comparative analysis of the laravel and codeigniter frameworks: For the implementation of the management system of merit and opposition competitions in the State University Pennsula de Santa Elena,"2018 13th Iberian Conference on Information Systems and Technologies (CISTI), Caceres, 2018, pp. 1-6.

[3] Huijie Su, Bo Cheng, Tong Wu and Xiaofeng Li, "Mashup service release based on SOAP and REST," Proceedings of 2011 International Conference on Computer Science and Network Technology, Harbin, 2011, pp. 1091-1095

[4] Y. Shulin and H. Jieping, "Research and implementation of Web Services in Android network communication framework Volley," 2014 11th International Conference on Service Systems and Service Management (ICSSSM), Beijing, 2014, pp. 1-3.

[5] M. M. Patil, A. Hanni, C. H. Tejeshwar and P. Patil, "A qualitative analysis of the performance of MongoDB vs MySQL database based on insertion and retriewal operations using a web/android application to explore load balancing Sharding in MongoDB and its advantages," 2017 International Conference on I-

SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC), Palladam, 2017, pp. 325-330.

[6] C. Gyrdi, R. Gyrdi, G. Pecherle and A. Olah, "A comparative study: MongoDB vs. MySQL," 2015 13th International Conference on Engineering of Modern Electric Systems (EMES), Oradea, 2015, pp. 1-6.

[7] S. Rautmare and D. M. Bhalerao, "MySQL and NoSQL database comparison for IoT application," 2016 IEEE International Conference on Advances in Computer Applications (ICACA), Coimbatore, 2016, pp. 235-238.

[8] Plekhanova, Julia. "Evaluating web development frameworks: Django, Ruby on Rails and CakePHP." Institute for Business and Information Technology (2009).

[9] Forcier, J., Bissex, P. and Chun, W.J., 2008. Python web development with Django. Addison-Wesley Professional.

[10] Chou, J., Chen, L., Ding, H., Tu, J., & Xu, B. (2013). A Method of Optimizing Django Based on Greedy Strategy. 2013 10th Web Information System and Application Conference.

[11] Rubio D. (2017) REST Services with Django. In: Beginning Django. Apress, Berkeley, CA

[12] L. Li and W. Chou, "Designing Large Scale REST APIs Based on REST Chart," 2015 IEEE International Conference on Web Services, New York, NY, 2015, pp. 631-638.

[13] L. Li, W. Chou, W. Zhou and M. Luo, "Design Patterns and Extensibility of REST API for Networking Applications," in IEEE Transactions on Network and Service Management, vol. 13, no. 1, pp. 154-167, March 2016.

[14] M. Lachgar, H. Benouda and S. Elfirdoussi, "Android REST APIs: Volley vs Retrofit," 2018 International Symposium on Advanced Electrical and Communication Technologies (ISAECT), Rabat, Morocco, 2018, pp. 1-6.

[15] Y. Shulin and H. Jieping, "Research and implementation of Web Services in Android network communication framework Volley," 2014 11th International Conference on Service Systems and Service Management (ICSSSM), Beijing, 2014, pp. 1-3.

[16] ASTROSAT Handbook Version 1.11, 2018, Dipankar Bhattacharya, Gulab C Dewangan, IUCAA. http://www.iucaa.in/~astrosat/AstroSat_handbook.pdf

[17] Hanisch, R. J., Farris, A., Greisen, E. W., Pence, W. D., Schlesinger, B. M., Teuben, P. J., Warnock, A. (2001). Definition of the Flexible Image Transport System (FITS). Astronomy & Astrophysics, 376(1), 359380.

[18] Inter-University Centre for Astronomy and Astrophysics, Pune, India. https://www.iucaa.in/

[19] ASTROSAT-CZTI Data Quality Reports http://www.iucaa.in/~astrosat/czti_dqr/

[20] Q. Mahmoud, I. Andrusiak and M. Altaei, "Toward a Reliable Service-Based Approach to Software Application Development," 2018 IEEE 20th Conference on Business Informatics (CBI), Vienna, 2018, pp. 168-177.

[21] Y. Shulin and H. Jieping, "Research and implementation of Web Services in Android network communication framework Volley," 2014 11th International Conference on Service Systems and Service Management (ICSSSM), Beijing, 2014, pp. 1-3.

[22] H. Hao, C. Zhou and T. Wang, "An implementation of Web Services on mobile terminal," 2013 6th International Conference on Information Management, Innovation Management and Industrial Engineering, Xi'an, 2013, pp. 175-178.

[23] M. L. Pandini, Z. Arifin and D. M. Khairina, "Design web service academic information system based multiplatform," 2014 The 1st International Conference on Information Technology, Computer, and Electrical Engineering, Semarang, 2014, pp. 297-302.

[24] Rubio D. (2017) REST Services with Django. In: Beginning Django. Apress, Berkeley, CA

[25] Huijie Su, Bo Cheng, Tong Wu and Xiaofeng Li, "Mashup service release based on SOAP and REST," Proceedings of 2011 International Conference on Computer Science and Network Technology, Harbin, 2011, pp. 1091-1095

[26] Z. Li and G. Wu, "Optimizing VM live migration strategy based on migration time cost modeling," 2016 ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS), Santa Clara, CA, 2016, pp. 99-109.

[27] Y. S. Wijaya and A. AkhmadArman, "A Framework for Data Migration Between Different Datastore of NoSQL Database," 2018 International Conference on ICT for Smart Society (ICISS), Semarang, 2018, pp. 1-6.

[28] S. Rautmare and D. M. Bhalerao, "MySQL and NoSQL database comparison for IoT application," 2016 IEEE International Conference on Advances in Computer Applications (ICACA), Coimbatore, 2016, pp. 235-238.

[29] Sample DQR. http://www.iucaa.in/~astrosat/czti_dqr/20151006_P01_003T01_9000000002_level2_00120/index.html

[30] The Open Group Base Specifications Issue 7, 2018 edition IEEE Std 1003.1-2017 (Revision of IEEE Std 1003.1-2008) http://pubs.opengroup.org/onlinepubs/9699919799/

[31] Yu Ping, Hu Hong-Wei, & Zhou Nan. (2014). Design and implementation of a MySQL database backup and recovery system. Proceeding of the 11th World Congress on Intelligent Control and Automation.

[32] COMPARISON PLAN FOR DATA WAREHOUSE SYSTEM ARCHITECTURES Abdolreza Hajmoosaei, Mehdi Kashfi, Punitha Kailasam School of Computing, Taylors University, PO Box 50603, Selangor, Malaysia.

[33] Information Resource Management and Retrieval Method in the Multi-Database Environment that Contains Image Database Mitsuaki Tsunakawa, Takashi Hoshino, Hiroki Machihara NTT Information and Communication Systems Laboratories 1-1 Hikari-no-oka, Yokosuka-Shi, Kanagawa, 239-0847, Japan

[34] Scientific Workflow Management by Database Management Y. Anastassia Ailamaki, Yannis E. Ioannidisz, Miron Livny. Department of Computer Sciences, University of Wisconsin, Madison, WI 53706 fnatassa,yannis,mirong@cs.wisc.edu

[35] Chou, J., Chen, L., Ding, H., Tu, J., & Xu, B. (2013). A Method of Optimizing Django Based on Greedy Strategy. 2013 10th Web Information System and Application Conference.

[36] C. Zhou and Q. Fei, "Warehouse Management System Development Base on Open Source Web Framework," 2016 International Conference on Industrial Informatics - Computing Technology, Intelligent Technology, Industrial Information Integration (ICIICII), Wuhan, 2016, pp. 65-68.

[37] Plekhanova, J., 2009. Evaluating web development frameworks: Django, Ruby on Rails and CakePHP. Institute for Business and Information Technology.

[38] Forcier, J., Bissex, P. and Chun, W.J., 2008. Python web development with Django. Addison-Wesley Professional.

[39] C. Tang, H. Bagheri, S. Paisarnsrisomsuk and K. Sullivan, "Towards Designing Effective Data Persistence through Tradeoff Space Analysis," 2017 IEEE/ACM 39th International Conference on Software Engineering Companion (ICSE-C), Buenos Aires, 2017, pp. 353-355.

[40] M. Lachgar, H. Benouda and S. Elfirdoussi, "Android REST APIs: Volley vs Retrofit," 2018 International Symposium on Advanced Electrical and Communication Technologies (ISAECT), Rabat, Morocco, 2018, pp. 1-6.

[41] Li Bo, Zhang Guoguang and Lv Xiangfen, "Presolving techniques in linear programming model," 2008 27th Chinese Control Conference, Kunming, 2008, pp. 22-25.

[42] J. Worrell, "Real-Time Model Checking: Algorithms and Complexity," 2008 15th International Symposium on Temporal Representation and Reasoning, Montreal, QC, 2008, pp. 19-19.