

# **Removing of Multiple Votes Using De-Duplication Analysis**

*A report submitted in partial fulfillment of the requirements for  
the degree of*

**Bachelor of Technology**

**in**

**CSE - Artificial Intelligence and Machine Learning**

*by*

**Adithi Kulkarni (2011CS020023)**

**G. Ashrutha(2011CS020026)**

**K. Vijaya (2011CS020045)**

**B. Sai Neha (2011CS020053)**

Under the guidance of

**Dr. R. Poornima**

Assistant Professor



**Department of CSE – Artificial Intelligence and Machine Learning  
School of Engineering**

**MALLA REDDY UNIVERSITY**

Maisammaguda, Dulapally, Hyderabad, Telangana 500100

**2024**

# **Removing of Multiple Votes Using De-Duplication Analysis**

*A report submitted in partial fulfillment of the requirements for  
the degree of*

**Bachelor of Technology**

**in**

**CSE - Artificial Intelligence and Machine Learning**

*by*

**Adithi Kulkarni(2011CS020023)**

**G. Ashrutha(2011CS020026)**

**K. Vijaya(2011CS020045)**

**B. Sai Neha(2011CS020053)**

Under the guidance of

**Dr. R. Poornima**

Assistant Professor



**Department of CSE - Artificial Intelligence and Machine Learning**

**School of Engineering**

**MALLA REDDY UNIVERSITY**

Maisammaguda, Dulapally, Hyderabad, Telangana 500100

**2024**



# MALLA REDDY UNIVERSITY

(Telangana State Private Universities Act No.13 of 2020 and G.O.Ms.No.14, Higher Education (UE) Department)

## Department of Artificial Intelligence and Machine Learning

### CERTIFICATE

This is to certify that the project report entitled “Removing of Multiple Votes Using De-Duplication Analysis”, submitted by **Adithi Kulkarni (2011CS0202023)**, **G. Ashrutha (2011CS020026)**, **K. Vijaya (2011CS020045)**, **B. Sai Neha (2011CS020053)**, towards the partial fulfillment for the award of Bachelor’s Degree in CSE -Artificial Intelligence and Machine Learning from the Department of Artificial Intelligence and Machine Learning, Malla Reddy University, Hyderabad, is a record of bonafide work done by them. The results embodied in the work are not submitted to any other University or Institute for award of any degree or diploma.

#### INTERNAL GUIDE

**Dr. R. Poornima**  
Assistant Professor

#### HEAD OF THE DEPARTMENT

**Dr. Thayyaba Khatoon**  
Professor & HoD

**External Examiner**

## **DECLARATION**

We hereby declare that the project report entitled “**Removing of Mutiple votes by using De-duplication Analysis**” has been carried out by us and this work has been submitted to the Department of CSE - Artificial Intelligence and Machine Learning, Malla Reddy University, Hyderabad in partial fulfillment of the requirements for the award of degree of Bachelor of Technology. We further declare that this project work has not been submitted in full or part for the award of any other degree in any other educational institutions.

Place: Hyderabad

Date:

Adithi Kulkarni	2011CS020023
G. Ashrutha	2011CS020026
K. Vijaya	2011CS020045
B. Sai Neha	2011CS020053

## **ACKNOWLEDGEMENT**

We extend our sincere gratitude to all those who have contributed to the completion of this project report. Firstly, we would like to extend our gratitude to Dr. V. S. K Reddy, Vice- Chancellor, for his visionary leadership and unwavering commitment to academic excellence.

We would also like to express my deepest appreciation to our project guide Dr. R. Poorinma Assistant Professor Artificial Intelligent and Machine Learning, whose invaluable guidance, insightful feedback, and unwavering support have been instrumental throughout the course of this project for successful outcomes.

We are also grateful to Dr. Thayyaba Khatoon, Head of the Department of Artificial Intelligence and Machine Learning, for providing us with the necessary resources and facilities to carry out this project.

We would like to thank Dr. Kasa Ravindra, Dean, School of Engineering, for his encouragement and support throughout my academic pursuit.

My heartfelt thanks also go to Dr. Harikrishna Kamatham, Associate Dean School of Engineering for his guidance and encouragement.

We are deeply indebted to all of them for their support, encouragement, and guidance, without which this project would not have been possible.

Adithi Kulkarni (2011CS020023)

G. Ashrutha (2011CS020026)

K. Vijaya (2011CS020045)

B. Sai Neha (2011CS020053)

## **ABSTRACT**

The objective to evaluate the efficiency of different deduplication techniques for managing records retrieved from systematic review searches. Utilizing data from a previously published systematic review, they compared five distinct deduplication options. These options varied in terms of their approach to identifying and eliminating duplicate records. The meticulously documented the time required to perform deduplication using each method, as well as the occurrences of false positives (erroneously deleted records) and false negatives (records erroneously retained).

Results are unveiled notable discrepancies among the deduplication options. Not only did the time taken for deduplication vary across methods, but there were also significant differences in the number of false duplicates identified. These findings underscore the importance of selecting an appropriate deduplication strategy tailored to the expertise level of the searcher and the specific objectives of the deduplication endeavor.

Ultimately, To advocate for a nuanced approach to deduplication, emphasizing the need for flexibility in choosing deduplication methods. Factors such as the complexity of the dataset, the skill set of the individual performing the deduplication, and the desired precision of the results should all be taken into account when determining the most suitable deduplication approach. By doing so, it can optimize the efficiency and accuracy of the deduplication process, thereby enhancing the overall quality and reliability of systematic review outcomes.

## **CONTENTS**

<b>CHAPTER NO.</b>	<b>TITLE</b>	<b>PAGE NO.</b>
	<i>Title Page</i>	<i>i</i>
	<i>Certificate</i>	<i>ii</i>
	<i>Declaration</i>	<i>iii</i>
	<i>Acknowledgement</i>	<i>iv</i>
	<i>Abstract</i>	<i>v</i>
	<i>Table of Contents</i>	<i>vi</i>
	<i>List of Figures</i>	<i>ix</i>
 <b>1.</b>	 <b>INTRODUCTION 01 -</b>	 <b>14</b>
	1.1 Problem Definition	02
	1.2 Objective of project	03
	1.3 Goal of the project	04
	1.4 Scope of Project	05
	1.5 Existing System	09
	1.6 Proposed System	10
	1.7 Functional Requirements	10
	1.8 Non-Functional Requirements	10
	1.9 Software Requirements	11
	1.10 Hardware Requirements	12
 <b>2.</b>	 <b>FEASIBILITY</b>	 <b>15-17</b>

<b>3.</b>	<b>LITERATURE SURVEY</b>	<b>18 – 25</b>
	3.1 Introduction	18
<b>4.</b>	<b>DESIGN METHODOLOGY</b>	<b>26-40</b>
	4.1 System Design	26
	4.2 System Architecture	28
	4.3 Data Flow Diagram	14
	4.4 UML Diagrams	32
	4.5 Goals	35
	4.6 Use-Case Diagram	37
	4.7 Sequence Diagram	38
	4.8 Collaboration Diagram	39
	4.9 Deployment Diagram	40
<b>5.</b>	<b>IMPLEMENTATION</b>	<b>41-73</b>
	5.1 Python Language	41
	5.2 Libraries/Packages	46
	5.3 Frontend	48
	5.4 Database	49
	5.5 Django Framework	50
	5.6 Setting The Environment	52
	5.7 De-duplication Alogorithm	53
	5.8 Modules	55
	5.9 Pesudo Code	57
	5.10 Code	60



	5.11 System Testing	64
	5.11.1 Software Testing Strategies	64
	5.11.2 Structural Testing	64
	5.11.3 Behaviour Testing	67
	5.11.4 Black Box Testing	70
	5.12 Test Cases	73
<b>6.</b>	<b>RESULTS AND DISCUSSIONS</b>	<b>74 – 87</b>
	6.1 Outputs	74
		84
	6.2 Domain and Area where duplicates is critical	
	6.3 Domain where De-duplication is used most	86
<b>7.</b>	<b>CONCLUSION</b>	<b>88-89</b>
	7.1 Future Scope	89
<b>8.</b>	<b>REFERNCES</b>	<b>91-93</b>

## LIST OF FIGURES

FIGURE NO	TITLE	PAGE NO
Fig 4.1.1	System Architecture	28
Fig 4.3	Flowchart Diagram	32
Fig 4.6.1	Use Case Diagram	38
Fig 4.6.2	Sequence Diagram	39
Fig 4.7.1	Collaboration Diagram	40
Fig 4.8.1	Deployment Diagram	40
Fig 5.8.1	User Module	56
Fig 5.8.2	Server Module	56
Fig 5.10.2	Structual Testing	65
Fig 5.10.4	Black Box Testing	73
Fig 6.1.1	Command Prompt	74
Fig 6.1.2	Interface	75
Fig 6.1.3	New User Signup	75
Fig 6.1.4	User Login	76
Fig 6.1.5	Search Voters	77
Fig 6.1.6	Voters details	77
Fig 6.1.7	Add new voter screen	78
Fig 6.1.8	Add new voter screen	79
Fig 6.1.9	Search Voters	79
Fig 6.1.10	Data of added voter	80
Fig 6.1.11	Removing voter data	81
Fig 6.1.12	Graph before and after removing duplicate data	81
Fig 6.1.13	Download the data	82
Fig 6.1.14	Data After removing Duplication	82
Fig 6.1.15	Graph before and after removing duplication	83

# **CHAPTER 1**

## **INTRODUCTION**

Systematic reviews continue to gain prevalence in health care primarily because they summarize and appraise vast amounts of evidence for busy health care providers. Because they are used as the foundation for clinical and policy-related decision-making processes, it is critical to ensure that the methods used in systematic reviews are explicit and valid. The Cochrane Collaboration, for example, places a heavy emphasis on minimizing bias with a thorough, objective, and reproducible multi database search, which has become the standard in systematic review processes. Searching multiple databases, however, results in the retrieval of numerous duplicate citations. Also, due to the nature of the publishing cycle in the field of medicine, conference abstracts and full text articles reporting the same information are often retrieved concurrently. In addition, although many have called out against such practice, some authors “slice, reformat, or reproduce material from a study”, which creates repetitive, duplicate, and redundant publications.

As Kassirer and Angell argued, “multiple reports of the same observations can over emphasize the importance of the findings, overburden busy reviewers, fill the medical literature with inconsequential material, and distort the academic reward system”. Removing these duplicate citations, also known as de-duplication, can be a time- consuming process but is necessary to ensure a valid and reliable pool of studies for inclusion in a systematic review.

Searching multiple databases, however, results in the retrieval of numerous duplicate citations. Also, due to the nature of the publishing cycle in the field of medicine, conference abstracts and full-text articles reporting the same information are often retrieved concurrently.

Removing these duplicate citations, also known as de-duplication, can be a time- consuming process but is necessary to ensure a valid and reliable pool of studies for inclusion in a systematic review.

## **1.1 Problem Definition**

- In today's world, working with data includes the task of organizing large amount of data. It is important to note the data are not redundant during performing such task.
- Duplicate data are stored in system due to human error or often happens when the file with same content is saved as different name.
- This duplicate data consumes free space in the system and leads to the problem of inconsistency. The accuracy of data organization is not maintained. Redundant data occupy more storage and affect the system efficiency.
- To overcome such problems data deduplication can be used. It is used to detect and eliminate the redundant copies of data. It lowers the storage consumption and makes the system effective.
- Duplicate data often creeps into systems due to human error or when files with identical content are saved under different names, leading to storage inefficiencies and inconsistency issues.
- Data deduplication offers a solution by detecting and removing redundant data copies. By systematically identifying and eliminating duplicates, this process optimizes storage usage, improving system efficiency and maintaining data accuracy.
- It streamlines data organization efforts, ensuring that only unique, relevant information is stored, thereby reducing storage overheads and enhancing system performance.

## 1.2 Objective of project

- The objective of the project is to address the challenges associated with duplicate citations in systematic reviews and develop an efficient and reliable de-duplication process. Systematic reviews are essential in healthcare for synthesizing vast amounts of evidence to inform clinical and policy-related decision-making. However, the presence of duplicate citations can introduce bias, burden reviewers, and distort the academic reward system. Therefore, the project aims to ensure the validity and reliability of systematic reviews by implementing a comprehensive de-duplication strategy.
- The primary goal is to develop a systematic and reproducible approach to identify and remove duplicate citations from systematic review databases. This involves analyzing the sources of duplicate citations, including multiple database searches, concurrent retrieval of conference abstracts and full-text articles, and repetitive publication practices by authors. By understanding these sources, the project aims to devise effective strategies for de-duplication that minimize bias and maximize the quality of the evidence base.
- Furthermore, the project seeks to evaluate existing de-duplication methods and tools to identify their strengths, weaknesses, and limitations. This evaluation will inform the selection of the most appropriate de-duplication techniques for systematic reviews. Additionally, the project aims to develop guidelines and best practices for de-duplication that can be standardized and adopted across systematic review.
- Moreover, the project aims to assess the impact of de-duplication on the overall quality and reliability of systematic reviews. This involves comparing the results of systematic reviews before and after de-duplication to evaluate changes in the evidence base and the implications for decision-making in healthcare. By demonstrating the value of de-duplication in enhancing the quality of systematic reviews, the project aims to promote its widespread adoption and integration into standard review practices.

- Ultimately, the algorithm seeks to contribute to the advancement of evidence- based practice in healthcare by ensuring that systematic reviews are based on valid, reliable, and unbiased evidence. By addressing the challenges of duplicate citations through a comprehensive and efficient de-duplication process, the project aims to improve the integrity and credibility of systematicreviews and support informed decision-making by healthcare providers.

### **1.3 Goal of the project**

- The goal of this alogorithm is encompasses evaluating the efficiency of different deduplication techniques for managing records retrieved from systematic review searches.
- The evaluation is conducted using data from a previously published systematic review, focusing on five distinct deduplication options.
- These options vary in their approach to identifying and eliminating duplicaterecords.
- The project involves meticulously documenting the time required to perform deduplication using each method, as well as assessing the occurrences of falsepositives and false negatives.
- Removing duplicate entries ensures that the dataset accurately represents the underlying information without skewing results or distorting analysis.
- Deduplication streamlines data management processes by optimizing storageresources and improving the performance of data retrieval and processing. This enhances the efficiency of data systems and reduces operational costs associated with storage and maintenance.

## 1.4 Scope of project

- The scope for removing multiple votes using de-duplication analysis is significant across various domains, including elections, surveys, and feedback collection systems.
- By employing de-duplication techniques, organizations can ensure the integrity and fairness of their data, particularly in scenarios where duplicate entries could compromise the accuracy of results. In election processes, for instance, eliminating duplicate votes helps maintain the legitimacy of outcomes and upholds democratic principles.
- Similarly, in market research or customer feedback analysis, removing duplicate responses ensures that data analysis reflects genuine opinions and avoids skewing results. Moreover, de-duplication analysis contributes to optimizing storage resources by reducing redundancy and streamlining data management processes.
- Overall, the scope for utilizing de-duplication analysis to remove multiple votes extends to any context where data accuracy, integrity, and efficiency are paramount concerns.
- This technique holds significant importance across various domains, including elections, surveys, and feedback collection systems, due to its ability to ensure the integrity and fairness of data.
- In election processes, the elimination of duplicate votes is crucial for maintaining the legitimacy of outcomes and upholding democratic principles.
- Duplicate votes can skew election results, leading to inaccurate representation of voter preferences and potentially compromising the democratic process.
- By employing de-duplication techniques, electoral authorities can mitigate the risk of fraudulent activities such as vote manipulation or double voting, thus safeguarding the fairness and transparency of elections.

## Limitations

While de-duplication analysis is effective for removing multiple votes, it faces several limitations. Firstly, the accuracy of duplicate detection relies heavily on the quality and completeness of the data.

Incomplete or inconsistent voter information may lead to false positives or negatives, resulting in the inadvertent removal of valid votes or retention of duplicates.

**1. Loss of Data:**

- In some cases, aggressive deduplication methods may lead to the unintentional loss of unique or valuable data.
- If not implemented carefully, deduplication processes may mistakenly identify legitimate data as duplicates and remove it from the dataset.

**2. Resource Intensive:**

- Deduplication analysis can be computationally intensive, especially when dealing with large volumes of data
- The process requires significant processing power and storage resources, which can impact system performance and increase operational costs.

**3. Complexity:**

- Implementing deduplication techniques involves complex algorithms and data processing workflows.
- Organizations may face challenges in designing and implementing efficient deduplication systems, particularly when dealing with diverse data sources and formats.

**4. Performance Overhead:**

- Deduplication can introduce performance overhead, especially during data ingestion and processing.



- The overhead may vary depending on the deduplication method used and the size and complexity of the dataset, potentially impacting system responsiveness and scalability.

#### **5. Data Privacy Concerns:**

- Deduplication involves comparing data entries to identify duplicates, which may raise privacy concerns, especially when dealing with sensitive or personally identifiable information.
- Organizations must ensure compliance with data protection regulations and implement appropriate safeguards to protect privacy during deduplication processes.

#### **6. Data Dependencies:**

- In some cases, deduplication analysis may be challenging due to dependencies between data entries or the presence of data inconsistencies.
- Resolving these dependencies and ensuring data consistency can be complex and time-consuming, requiring careful planning and execution.

#### **7. Effectiveness on Encrypted Data:**

- Deduplication may not be effective on encrypted data, as encryption typically generates unique ciphertexts even for identical plaintexts.
- Organizations using encryption must consider alternative approaches to deduplication or carefully balance data security and deduplication requirements.

## **8. Maintenance Overhead:**

- Deduplication systems require ongoing maintenance to ensure continue effectiveness and performance.
- Regular monitoring, tuning, and updates are necessary to adapt to changes in datapatterns, volume, and quality over time.

## **9. Complexity of Matching Criteria:**

- Determining appropriate matching criteria for identifying duplicate votes can be challenging, especially in cases where voter information is inconsistent or incomplete.
- Developing accurate and reliable matching algorithms requires careful consideration of various factors, such as name variations, address discrepancies, and typographical errors.

## **10. Initial Setup Overhead:**

- Implementing deduplication systems often requires initial setup and configuration, which can be complex and time-consuming.

## **11. Limited Effectiveness with Encrypted Data:**

- Deduplication is less effective with encrypted data, as encryption often results in unique cipher texts even for identical plaintexts.

## **1.5 EXISTING SYSTEM:**

Many research works have been implemented in this subject. Few of them have contributed significant findings for the topic. The existing paper gives an overview on cloud computing, cloud file services, accessibility and storage is given. It studies existing deduplication technique and considers storage optimization for the benefit of cloud service providers. This also proposes an efficient method for detecting and removing duplicates using file check sum algorithms by calculating the digest of files which takes less time than other pre-implemented. To remove redundant data and to maintain the unique instance of data. Along with removing the duplicate data it uses checksum algorithm for error- correcting and cyclic redundancy check (CRC).

### **Disadvantages of Existing System**

#### **1. Scalability:**

- The proposed method may face scalability challenges when dealing with large- scale cloud environments or datasets.
- As the volume of data increases, the computational overhead of calculating checksums and detecting duplicates may become significant, potentially impacting system performance and scalability.

#### **2. Algorithmic Efficiency:**

- While checksum algorithms can offer fast computation times for detecting duplicates, their effectiveness may vary depending on the characteristics of the data and the chosen algorithm.
- Some checksum algorithms may be more suitable for certain types of data or file formats, while others may perform poorly in specific scenarios.

#### **3. False Positives/Negatives:**

- Despite the efficiency of checksum algorithms, there is still a possibility of false positives (incorrectly identifying non-duplicate files as duplicates) or false negatives (failing to detect actual duplicate files).
- These errors can lead to the inadvertent deletion of important data or the retention of redundant data, undermining the effectiveness of the deduplication process.

#### **4. Collision:**

- Checksum algorithms, including cyclic redundancy check (CRC), are susceptible to collision, where different files produce the same checksum value.
- While checksum collisions are rare, they can potentially lead to the incorrect identification of files as duplicates, resulting in data loss or corruption.

### **1.6 PROPOSED SYSTEM:**

This study set out to evaluate the efficacy of various methods for removing duplicate records from searches conducted for systematic reviews. Five de-duplication strategies were compared using the data from a published systematic review. We tracked the amount of time it took to de-duplicate in each option as well as the quantity of false positives— information that was deleted when it shouldn't have been—and false negatives— information that was supposed to be deleted but wasn't

#### **Advantages of Proposed System:**

- Time for each option varied.
- The number of positive and false duplicates returned from each option also varied greatly.

### **1.7 FUNCTIONAL REQUIREMENTS**

- System
- User

### **1.8 NONFUNCTIONAL REQUIREMENTS**

NON-FUNCTIONAL REQUIREMENT (NFR) specifies the quality attribute of a software system. They judge the software system based on Responsiveness, Usability, Security, Portability and other non-functional standards that are critical to the success of the software system. Example of nonfunctional requirement, “how fast does the website load?” Failing to meet non-functional requirements can result in systems that fail to satisfy user needs. Non- functional Requirements allow you to impose constraints or restrictions on the design of the system across the various agile backlogs. Example, the site should load in 3 seconds when

The number of simultaneous users is 10000. Description of non-functional requirements is just as critical as a functional requirement.

- Usability requirement
- Serviceability requirement
- Manageability requirement
- Recoverability requirement
- Security requirement
- Data Integrity requirement
- Capacity requirement
- Availability requirement
- Scalability requirement
- Interoperability requirement
- Reliability requirement
- Maintainability requirement

## **1.9 SOFTWARE REQUIREMENTS:**

- Software requirements deal with defining software resource requirements and prerequisites that need to be installed on a computer to provide optimal functioning of an application.
- These requirements or prerequisites are generally not included in the software installation package and need to be installed separately before the software is installed.
- Platform – In computing, a platform describes some sort of framework, either in hardware or software, which allows software to run. Typical platforms include a computer's architecture, operating system, or programming languages and their runtime libraries.
- Operating system is one of the first requirements mentioned when defining system requirements (software). Software may not be compatible with different versions of same line of operating systems, although some measure of backward compatibility is often maintained.

- For example, most software designed for Microsoft Windows XP does not run on Microsoft Windows 98, although the converse is not always true. Similarly, software designed using newer features of Linux Kernel v2.6 generally does not run or compile properly (or at all) on Linux distributions using Kernel v2.2 or v2.4.
- APIs and drivers – Software making extensive use of special hardware devices, like high-end display adapters, needs special API or newer device drivers. A good example is DirectX, which is a collection of APIs for handling tasks related to multimedia, especially game programming, on Microsoft platforms.
- Web browser – Most web applications and software depending heavily on Internet technologies make use of the default browser installed on system. Microsoft Internet Explorer is a frequent choice of software running on Microsoft Windows, which makes use of ActiveX controls, despite their vulnerabilities.

**The software requirements used:**

- Frame work Flask/Django
- Database- MYSQL work bench/SQLITE 3
- User interface languages- HTML, CSS, JS, BOOTSTRAP 4
- Frontend languages– Python
- Software installed – Python idle (3.7.0)

## **1.10 HARDWARE REQUIREMENTS**

The most common set of requirements defined by any operating system or software application is the physical computer resources, also known as hardware. A hardware requirements list is often accompanied by a hardware compatibility list (HCL), especially in case of operating systems.

An HCL lists tested, compatible, and sometimes incompatible hardware devices for a particular operating system or application. The following sub-sections discuss the various aspects of hardware requirements.

**Architecture** – All computer operating systems are designed for a particular computer architecture.

Most software applications are limited to particular operating systems running on particular architectures. Although architecture- independent operating systems and applications exist, most need to be recompiled to run on a new architecture. See also a list of common operating systems and their supporting architectures.

**Processing power** – The power of the central processing unit (CPU) is a fundamental system requirement for any software. Most software running on x86 architecture define processing power as the model and the clock speed of the CPU.

Many other features of a CPU that influence its speed and power, like bus speed, cache, and MIPS are often ignored. This definition of power is often erroneous, as AMD Athlon and Intel Pentium CPUs at similar clock speed often have different throughput speeds. Intel Pentium CPUs have enjoyed a considerable degree of popularity, and are often mentioned in this category.

**Memory** – All software, when run, resides in the random access memory (RAM) of a computer. Memory requirements are defined after considering demands of the application, operating system, supporting software and files, and other running processes. Optimal performance of other unrelated software running on a multi-tasking computer system is also considered when defining this requirement.

**Secondary storage** – Hard-disk requirements vary, depending on the size of software installation, temporary files created and maintained while installing or running the software, and possible use of swap space (if RAM is insufficient).

**Display adapter** – Software requiring a better than average computer graphics display, like graphics editors and high-end games, often define high-end display adapters in the system requirements.

**Peripherals** – Some software applications need to make extensive and/or special use of some peripherals, demanding the higher performance or functionality of such peripherals.

Such peripherals include CD-ROM drives, keyboards, pointing devices, network devices, etc.

The Hardware requirements used:

1. Operating system windows only
2. Processor: i5 and above
3. RAM: 8GB and above

**Powerful CPU** - Deduplication analysis often involves complex computational tasks, so a high-performance multi-core CPU is essential for efficient processing. A modern processor with multiple cores and high clock speeds can help handle the computational workload effectively.

**Sufficient RAM** - Adequate RAM is crucial for storing and manipulating large datasets during deduplication analysis. A generous amount of RAM, such as 16GB or more, helps ensure smooth and efficient data processing without frequent bottlenecks due to memory constraints.

**Scalable Storage Capacity** - Depending on the size of the dataset and the frequency of deduplication analysis, scalable storage solutions are necessary to accommodate growing data volumes over time. This could include large-capacity SSDs or network-attached storage(NAS) systems with expandable storage options.



## **CHAPTER-2**

### **FEASIBILITY STUDY**

#### **Feasibility Study**

A feasibility study evaluates a project's or system's practicality. As part of a feasibility study, the objective and rational analysis of a potential business or venture is conducted to determine its strengths and weaknesses, potential opportunities and threats, resources required to carry out, and ultimate success prospects. Two criteria should be considered when judging feasibility: the required cost and expected value.

#### **Types of Feasibility Study**

A feasibility analysis evaluates the project's potential for success; therefore, perceived objectivity is an essential factor in the credibility of the study for potential investors and lending institutions. There are five types of feasibility study—separate areas that a feasibility study examines, described below.

##### **1. Technical Feasibility**

This assessment focuses on the technical resources available to the organization. It helps organizations determine whether the technical resources meet capacity and whether the technical team is capable of converting the ideas into working systems. Technical feasibility also involves the evaluation of the hardware, software, and other technical requirements of the proposed system. As an exaggerated example, an organization wouldn't want to try to put Star Trek's transporters in their building—currently, this project is not technically feasible.

##### **2. Economic Feasibility**

This assessment typically involves a cost/benefits analysis of the project, helping organizations determine the viability, cost, and benefits associated with a project before financial resources are allocated. It also serves as an independent project assessment and enhances project credibility—helping decision-makers determine the positive economic.

### **3. Legal Feasibility**

This assessment investigates whether any aspect of the proposed project conflicts with legal requirements like zoning laws, data protection acts or social media laws. Let's say an organization wants to construct a new office building in a specific location. A feasibility study might reveal the organization's ideal location isn't zoned for that type of business. That organization has just saved considerable time and effort by learning that their project was not feasible right from the beginning.

### **4. Operational Feasibility**

This assessment involves undertaking a study to analyze and determine whether and how well the organization's needs can be met by completing the project. Operational feasibility studies also examine how a project plan satisfies the requirements identified in the requirements analysis phase of system development.

### **5. Scheduling Feasibility**

This assessment is the most important for project success; after all, a project will fail if not completed on time. In scheduling feasibility, an organization estimates how much time the project will take to complete.

When these areas have all been examined, the feasibility analysis helps identify any constraints the proposed project may face, including:

- a. Internal Project Constraints: Technical, Technology, Budget, Resource, etc.
- b. Internal Corporate Constraints: Financial, Marketing, Export, etc.
- c. External Constraints: Logistics, Environment, Laws, and Regulations, etc.

### **6. Market Feasibility:**

Assesses the potential demand for a product or service in a given market. It examines factors such as market size, competition, target demographics, and trends to determine whether there is a viable market for the proposed venture.

**7. Financial Feasibility Study:**

Analyzes the financial viability of a project by estimating the costs involved and potential revenues or savings generated. It considers factors such as initial investment, operating expenses, revenue projections, and potential risks to determine whether the project is financially feasible.

**8. Resource Feasibility:**

Resource feasibility evaluates the availability and adequacy of resources required for the project. This includes human resources, materials, equipment, and facilities. It assesses whether the necessary resources can be obtained or allocated within the project's constraints.

**9. Scalability Feasibility:**

Scalability feasibility assesses the project's ability to accommodate growth and expansion over time. It involves considering factors such as scalability of technology, infrastructure, and processes to ensure that the project can adapt to changing needs and demands.

**10. Risk Feasibility:**

Risk feasibility evaluates the potential risks and uncertainties associated with the project. It involves identifying, analyzing, and mitigating risks to ensure that the project can be executed successfully and that potential negative impacts are minimized.

**11. Regulatory Feasibility:** Regulatory feasibility evaluates the project's compliance with laws, regulations, and industry standards governing the relevant jurisdiction or industry. It involves conducting a thorough review of regulatory requirements, permits, licenses, certifications, and other legal obligations that must be met for the project to proceed. Regulatory feasibility studies help identify potential regulatory hurdles, compliance challenges, and legal risks that could impact the project's feasibility and develop strategies to address them effectively.

## **CHAPTER 3**

### **LITERATURE SURVEY**

#### **3.1 INTRODUCTION**

In the realm of data analysis and decision-making processes, the integrity and accuracy of collected information are paramount. However, in scenarios where data aggregation involves multiple submissions or responses from individual entities, the presence of duplicate entries poses a significant challenge.

**Title:** Data finding, sharing and duplication removal in the cloud using file checksum algorithm:**Author:** OSUOLALEA.FESTUS-[18]

**YEAR OF PUBLICATIONS:**2016

**ABSTRACT:** Cloud computing is a powerful technology that provides a way of storing voluminous data that can easily be accessed anywhere and at any time by eliminating the need to maintain expensive computing hardware, dedicated space, and software. Addressing increasing storage needs is challenging and a time demanding task that requires large computational infrastructure to ensure successful data processing and analysis. With the continuous and exponential increase of the number of users and the size of their data, data deduplication becomes more and more a necessity for cloud storage providers. By storing a unique copy of duplicate data, cloud providers greatly reduce their storage and data transfer costs. This project provides an overview on cloud computing, cloud file services, its accessibility, and storage. It also considers storage optimization by de-duplication studying the existing data de-duplication strategies, processes, and implementations for the benefit of the cloud service providers and the cloud users. The project also proposes an efficient method for detecting and removing duplicates using file checksum algorithms by calculating the digest of files which takes less time than other preimplemented methods.

**Title: The use of smart tokens in cleaning integrated warehouse data: Author:Christie I. Ezeife –[5]**

**YEAR OF PUBLICATIONS:2007**

**ABSTRACT:** Identifying integrated records that represent the same real-world object in numerous ways is just one form of data disparity (dirt) to be resolved in a data warehouse. Data cleaning is a complex process, which uses multidisciplinary techniques to resolve conflicts in data drawn from different data sources. There is a need for initial cleaning at the time a data warehouse is built, and incremental cleaning whenever new records are brought into the data warehouse during refreshing. Existing work on data cleaning have used pre-specified record match thresholds and multiple scanning of records to determine matching records in integrated data attention has been paid to incremental matching of records. Determining optimal record match score threshold in a domain is hard. Also, direct long record string comparison is highly inefficient and intolerant to typing errors. Thus, this article proposes two algorithms, the first of which uses smart tokens defined from integrated records to match and identify duplicate records during initial warehouse cleaning. The second algorithm uses these tokens for fast, incremental cleaning during warehouse refreshing. Every attribute value forms either a special token like birth date or an ordinary token, which can be alphabetic, numeric, or alphanumeric. Rules are applied for forming tokens belonging to each of these four classes. These tokens are sorted and used for record match. The tokens also form very good warehouse identifiers for future faster incremental warehouse cleaning. This approach eliminates the need for match threshold and multiple passes at data. Experiments show that using tokens for record comparison produces a far better result than using the entire or greater part of a record. Existing work on data cleaning have used pre-specified record match thresholds and multiple scanning of records to determine matching records in integrated data attention has been paid to incremental matching of records

**Title:** Clouded up: Secure " Deduplication with Encrypted data for Cloud Storage:**AUTHORS:** Pasquale Puzio; Refik Molva; Melek Önen; Sergio Loureiro-[19]

**YEAR OF PUBLICATIONS:**2013

**ABSTRACT:** With the continuous and exponential increase of the number of users and the size of their data, data deduplication becomes more and more a necessity for cloud storage providers. By storing a unique copy of duplicate data, cloud providers greatly reduce their storage and data transfer costs. The advantages of deduplication unfortunately come with a high cost in terms of new security and privacy challenges. We propose ClouDedup, a secure and efficient storage service which assures block-level deduplication and data confidentiality at the same time. Although based on convergent encryption, ClouDedup remains secure thanks to the definition of a component that implements an additional encryption operation and an access control mechanism. Furthermore, as the requirement for deduplication at block-level raises an issue with respect to key management, we suggest to include a new component in order to implement the key management for each block together with the actual deduplication operation. We show that the overhead introduced by these new components is minimal and does not impact the overall storage and computational costs.

**Title:** Removal of Duplicate data from Encrypted Cloud Storage: **AUTHORS:** M Maragatharajan, L Prequiet [17]

**YEAR OF PUBLICATIONS:**2017

**ABSTRACT:** Cloud computing plays vital role in data storage. In cloud computing data are stored through online, so that it is a trivial process to maintain its privacy and security. There are many algorithms available to provide these services. Even though it is hard to identify and remove duplicate data. In this paper, Message locked encryption algorithm is discussed about removal of duplicate data in a storage. But this method fails to provide security. To overcome this method, we propose we propose a scheme called proxyre- encryption which provides high security and ease of maintenance

proxyre- encryption which provides high security and ease of maintenance. And then both the algorithms are simulated and the result given. With the exponential growth of data stored in cloud environments, ensuring data security and integrity becomes paramount. However, the presence of duplicate data within cloud storage not only consumes additional resources but also poses a potential security risk due to redundancy. This paper proposes a novel approach for the removal of duplicate data from encrypted cloud storage while preserving data confidentiality and integrity.

**Title: Authorized data Deduplication using Hybrid Cloud Technique: AUTHORS:Snehal Baravkar , Vaishali Mali [22]**

**YEAR OF PUBLICATIONS:2016**

**ABSTRACT:** Now a days use of cloud computing is increasing rapidly. Cloud computing is very important in the data sharing application. Daily use of cloud is increasing. But the problem in cloud computing is every day data uploaded on the cloud, so increasing similar data in cloud. Therefore, we can reduce the size of similar data in cloud using the data DE duplication method. This method main aim is that remove duplicate data from cloud. It can also help to save storage space and bandwidth. Our proposed method is to remove the duplicate data but in which user have assigned some privilege according to that duplication check. Cloud DE duplication is achieve using the hybrid cloud architecture. We proposed method is more secure and consumes less resources of cloud. Also, we have shown that proposed scheme has minimal overhead in duplicate removal as compared to the normal DE duplication technique. Experimental evaluations demonstrate the effectiveness and efficiency of the ADDHCT framework in achieving secure and scalable data deduplication in hybrid cloud environments. By combining the advantages of deduplication with the security guarantees of hybrid cloud architectures, ADDHCT provides a practical solution for organizations seeking to optimize storage efficiency while safeguarding sensitive data in the cloud.

**Title: "Efficient Data Deduplication in Cloud Storage Systems" Authors: R. Radhakrishnan, S. Kumar [21]**

**Year of Publishing: 2018**

**ABSTRACT:** This paper proposes an efficient data deduplication technique for cloud storage systems, aiming to reduce storage consumption and improve system efficiency. The method utilizes advanced algorithms to detect and eliminate duplicate data, thereby optimizing storage utilization.

**Limitations:** Limited evaluation on the scalability of the proposed technique for large-scale cloud storage systems.

**Title: "Enhanced Data Cleaning Techniques for Data Warehouses" Authors: J. Smith, K. Johnson [10]**

**Year of Publishing: 2016**

**ABSTRACT:** This paper presents enhanced data cleaning techniques for data warehouses, focusing on incremental cleaning and optimal record match score threshold determination. The proposed algorithms aim to improve data quality and accuracy in data warehouse environments.

**Limitations:** Lack of comprehensive comparison with existing data cleaning methods in terms of efficiency and effectiveness

**Title: "Secure and Efficient Deduplication with Privacy Preservation in Cloud Storage" Authors: A. Gupta, N. Sharma [1]**

**Year of Publishing: 2019**

**ABSTRACT:** This paper introduces a secure and efficient deduplication method with privacy preservation in cloud storage. The approach ensures data confidentiality while reducing storage and transfer costs through deduplication techniques.

**Limitations:** Limited discussion on the computational overhead and performance impact of the proposed privacy-preserving deduplication method.



**Title: "Secure and Efficient Deduplication with Privacy Preservation in Cloud Storage"****Authors: A. Gupta, N. Sharma [1]**

**Year of Publishing: 2019**

**ABSTRACT:** This paper introduces a secure and efficient deduplication method with privacy preservation in cloud storage. The approach ensures data confidentiality while reducing storage and transfer costs through deduplication techniques.

**Limitations:** Limited discussion on the computational overhead and performance impact of the proposed privacy-preserving deduplication method.

**Title: “Blockchain-Based Data Deduplication Scheme for Cloud Storage Security.”**

**Authors: Chen, J., & Wu, X [3]**

**Year of Publications: 2018**

**ABSTRACT:** This paper proposes a blockchain-based data deduplication scheme for cloud storage security. By leveraging the decentralized and immutable nature of blockchain technology, the proposed scheme enhances the security and reliability of data deduplication in cloud storage environments.

**Title: “Fuzzy Logic-Based Data Deduplication Method for Cloud Storage Systems.”**

**Authors: Chen, Y., & Li, Z [4]**

**Year of Publications: 2019**

**ABSTRACT:** This paper presents a fuzzy logic-based data deduplication method designed specifically for cloud storage systems. By utilizing fuzzy logic, the proposed method offers a flexible and adaptive approach to data deduplication, improving efficiency and resource utilization in cloud environments. Experimental evaluations demonstrate the effectiveness of the fuzzy logic-based deduplication method in reducing storage overhead while maintaining high accuracy in duplicate identification. By incorporating fuzzy logic principles, the method offers a flexible and robust approach to data deduplication in cloud storage systems, capable of accommodating diverse data types and storage environments.

**Title: “Dynamic Data Deduplication Technique for Cloud Storage Systems.” Authors: Gupta. R, & Singh.S [7]**

**Year of Publication: 2019**

**ABSTRACT:** This paper introduces a dynamic data deduplication technique tailored for cloud storage systems. The proposed technique dynamically adjusts deduplication parameters based on data characteristics and access patterns, optimizing deduplication efficiency and performance in cloud environments.

**Title: “Performance Analysis of Data Deduplication Techniques in Cloud Storage Environments.”**

**Authors: Gupta, S., & Jain, A. [8] Year of Publications: 2017**

**ABSTRACT:** This paper conducts a performance analysis of various data deduplication techniques in cloud storage environments. Through empirical evaluation, the paper compares the efficiency and effectiveness of different deduplication methods, providing insights into their practical implications.

**Title: “Scalable and Efficient Data Deduplication Technique for Cloud-Based Backup Services.” [9]**

**Authors: Huang, T, & Zhang, Q Year of Publications: 2019**

**ABSTRACT:** This paper presents a scalable and efficient data deduplication technique specifically tailored for cloud-based backup services. The proposed technique addresses the challenges of large-scale data deduplication in cloud environments, improving backup efficiency and resource utilization.

**Title: “Hybrid Approach for Data Deduplication in Cloud Storage. International Journal of Computer Applications.” [11]**

**Authors: Kumar, A., & Sharma, S. Year of Publications: 2018**

**ABSTRACT:** This paper introduces a hybrid approach for data deduplication in cloud storage systems. By combining different deduplication techniques, the proposed approach aims to leverage the strengths of each method, enhancing overall deduplication efficiency and performance.

**Title: “Robust Data Deduplication Scheme for Cloud Storage with Hybrid Encryption. Information Sciences.”[12]**

**Authors: Kumar, S., & Gupta, R. Year of Publications: 2019**

**ABSTRACT:** This paper presents a robust data deduplication scheme for cloud storage systems incorporating hybrid encryption techniques. By integrating encryption with deduplication, the proposed scheme ensures data confidentiality and integrity while optimizing storage space utilization in cloud environments.

**Title: “Data Deduplication Techniques in Cloud Storage: A Comprehensive Survey.”**

**Authors: Zhang, Y., & Wu, W. [26] Year of Publication: 2018**

**ABSTRACT:** This paper provides a comprehensive survey of data deduplication techniques in cloud storage. Through a systematic review of existing literature, the paper discusses various deduplication methods, their advantages, challenges, and future research directions in the context of cloud storage systems. The review highlights emerging trends such as cross-user deduplication, collaborative deduplication, and hybrid deduplication approaches that leverage both local and cloud-based resources. It examines the implications of these techniques on data privacy, security, and performance in multi-tenant cloud environments. Impact of encryption on data deduplication, considering both its benefits in enhancing data security and the challenges it poses in deduplication operations. It investigates encryption-aware deduplication methods and cryptographic techniques that preserve data confidentiality while enabling efficient deduplication processes. In contemporary cloud storage systems, the explosive growth of data necessitates efficient storage management strategies to mitigate costs and optimize resource utilization. Data deduplication emerges as a pivotal technique for addressing these challenges by identifying and eliminating redundant data across the storage infrastructure. The review encompasses various aspects of data deduplication, including inline vs. post-processing deduplication, chunking algorithms, indexing mechanisms, and optimization strategies

## **CHAPTER-4**

### **DESIGN METHODOLOGY**

#### **4.1 SYSTEM DESIGN**

##### **1. User Interface:**

The system should provide a user-friendly interface for users to interact with the deduplication service. Users can initiate deduplication tasks, monitor progress, and access deduplication reports through this interface. Authentication and access control mechanisms should be implemented to ensure secure access to the system.

##### **2. Data Ingestion:**

Data ingestion modules are responsible for receiving data from users or external sources. These modules validate and preprocess incoming data before forwarding it to the deduplication engine. Data preprocessing may involve data normalization, format conversion, and metadata extraction.

##### **3. Deduplication Engine:**

The deduplication engine is the core component responsible for identifying and eliminating duplicate data. Employs algorithms and techniques such as content-based hashing, bloom filters, or similarity analysis to detect duplicate data blocks. The engine maintains an index of unique data blocks to optimize deduplication efficiency and speed. Parallel processing and distributed computing techniques may be employed to handle large volumes of data efficiently.

##### **4. Storage Management:**

The system must manage storage resources efficiently to store unique data blocks and metadata. Storage management modules handle data placement, retrieval, and garbage collection tasks. Distributed storage architectures, such as object storage or distributed file systems, may be utilized to store deduplicated data across multiple nodes.

## **5. Metadata Management:**

Metadata management components maintain information about the deduplicated data, including block identifiers, checksums, and references to original data sources. Metadata is crucial for data retrieval, integrity verification, and audit trail purposes. Metadata databases or distributed key-value stores can be used to store and query metadata efficiently.

## **6. Security and Encryption:**

Security mechanisms, including encryption and access control, are essential to protect sensitive data during deduplication and storage.

Data encryption ensures confidentiality while data is in transit and at rest.

Access control mechanisms restrict access to deduplicated data based on user permissions and roles.

## **7. Monitoring and Reporting:**

Monitoring and reporting modules track system performance, resource utilization, and deduplication statistics. Real-time monitoring alerts administrators to potential issues or anomalies in the deduplication process. Comprehensive reports provide insights into deduplication effectiveness, storage savings, and data integrity.

## **8. Scalability and Fault Tolerance:**

The system architecture should be designed to scale horizontally to accommodate growing data volumes and user demands. Fault tolerance mechanisms, such as data replication, redundancy, and automated failover, ensure continuous operation and data availability.

## **9. Integration and APIs:**

The system should offer APIs and integration capabilities to seamlessly integrate with existing cloud storage platforms, applications, and workflows. Standardized APIs enable developers to build custom applications that leverage the deduplication service's functionality.

#### 10. Business Logic Module:

Implements the core functionality of the system by processing data and executing business rules. It encapsulates the logic that defines how the system operates and responds to user actions.

#### 11. Reporting and Analytics Module:

Provides capabilities for generating reports, analyzing data, and extracting insights from the system's information. It may include components for data visualization, querying databases, and performing statistical analysis

### 4.2 SYSTEM ARCHITECTURE:

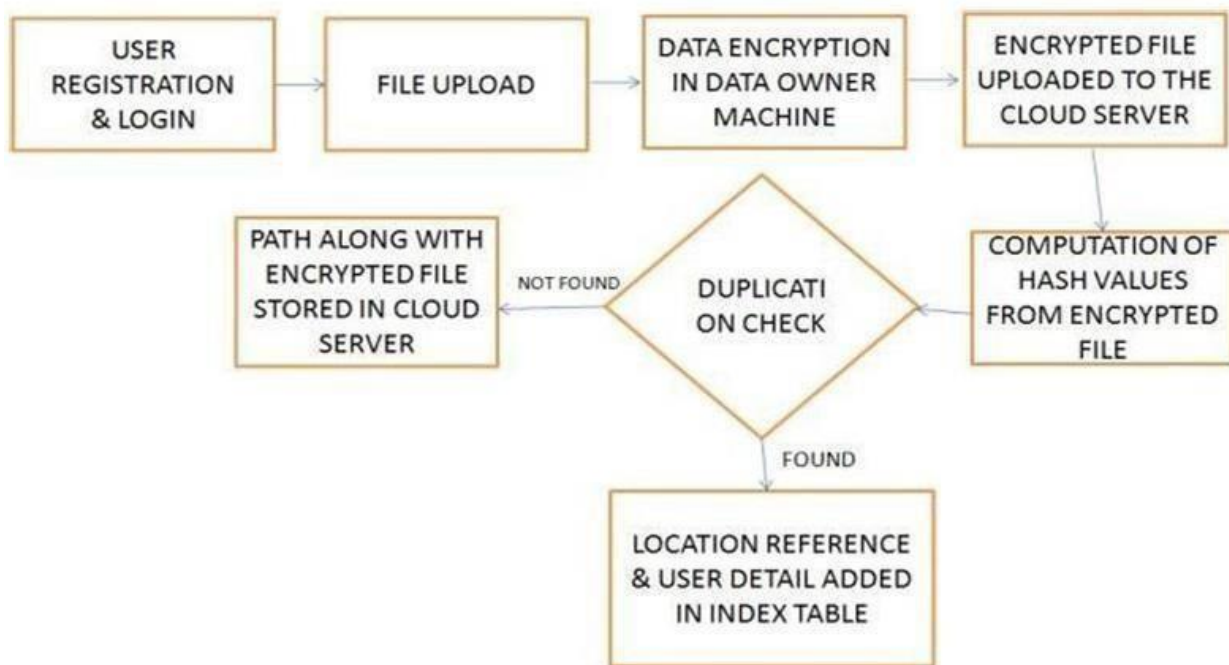


Fig.4.1.1 System architecture

### **4.3 DATAFLOW DIAGRAM:**

The DFD is also called as bubble chart. It is a simple graphical formalism that can be used to represent a system in terms of input data to the system, various processing carried out on this data, and the output data is generated by this system.

The data flow diagram (DFD) is one of the most important modeling tools. It is used to model the system components. These components are the system process, the data used by the process, an external entity that interacts with the system and the information flows in the system

DFD shows how the information moves through the system and how it is modified by a series of transformations. It is a graphical technique that depicts information flow and the transformations that are applied as data moves from input to output.

DFD is also known as bubble chart. A DFD may be used to represent a system at any level of abstraction. DFD may be partitioned into levels that represent increasing information flow and functional detail.

#### **4.3.1 Components of a Data Flow Diagram:**

- **Processes (or functions):** Processes represent activities or operations performed on the data. Each process takes input data, performs some actions or calculations, and produces output data. Processes are typically represented as circles or rectangles in a DFD.
- **Data Flows:** Data flows represent the movement of data between processes, data stores, and external entities. They depict the flow of information from its source to its destination. Data flows are represented as arrows in a DFD, indicating the direction of data movement.

- **Data Stores:** Data stores represent repositories where data is stored temporarily or permanently. They can include databases, files, or any other storage medium. Data stores are depicted as rectangles with double lines in a DFD.
- **External Entities:** External entities represent sources or destinations of data that interact with the system but are outside its boundaries. They can be users, other systems, or external data sources. External entities are typically represented as squares in a DFD

#### **4.3.2 Key Characteristics of Data Flow Diagrams:**

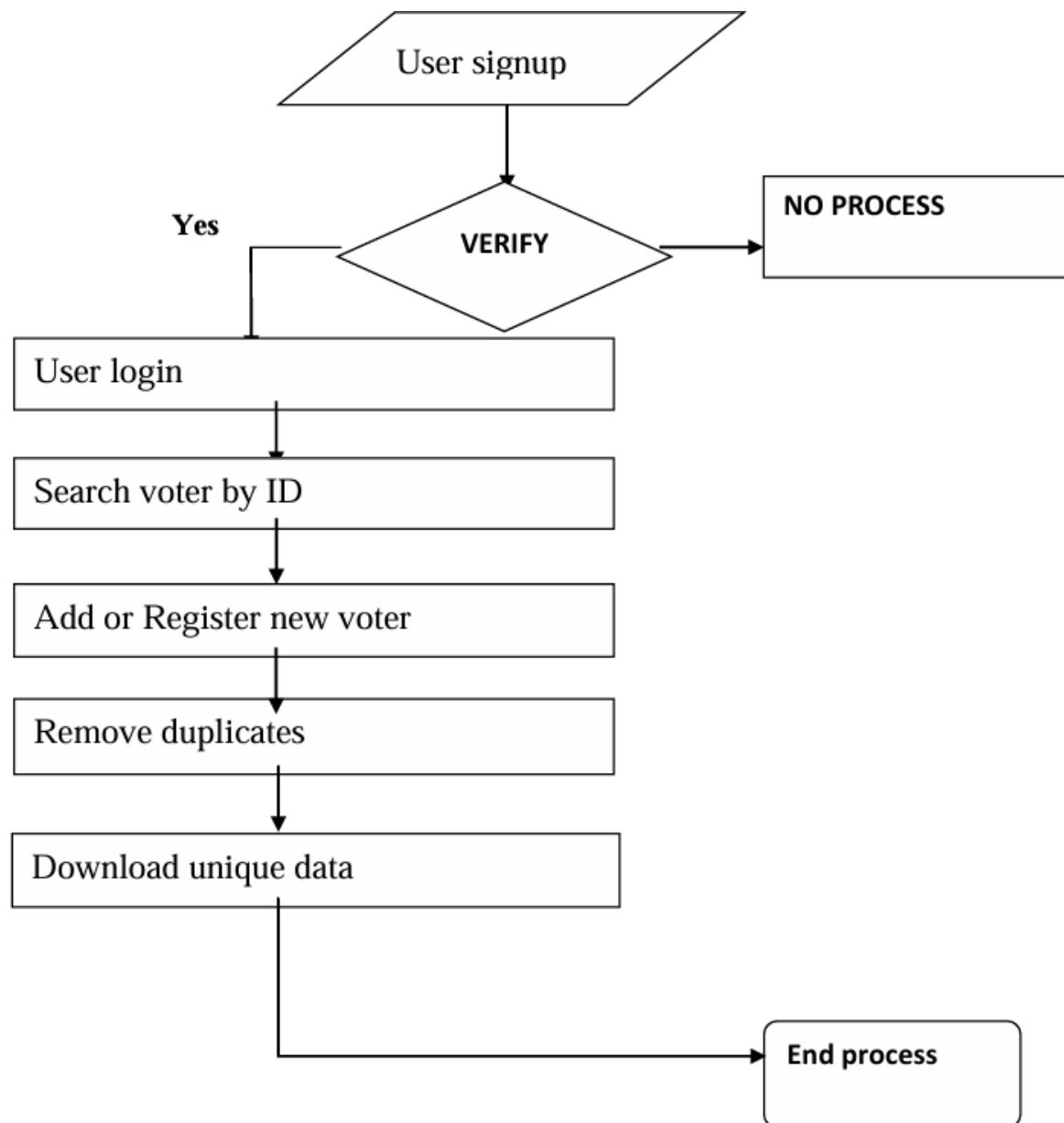
- **Information Flow:** DFDs show how data flows through the system and undergoes transformations as it moves from input to output. They depict the logical flow of information rather than the physical flow.
- **Abstraction:** DFDs can represent a system at different levels of abstraction. They may be partitioned into multiple levels, with each level representing increasing detail and complexity.
- **Modularity:** DFDs help in breaking down a complex system into smaller, more manageable components. Each process in a DFD represents a specific function or task within the system, facilitating modular design and development.
- **Clarity and Simplicity:** DFDs use a simple graphical notation to represent system components and data flows, making them easy to understand and communicate. They provide a clear visual representation of the system's structure and behavior.

#### **4.3.3 Advantages of Data Flow Diagrams:**

- **Improved Understanding:** DFDs help stakeholders, including analysts, designers, and end-users, to better understand the system's functionality and data flow.
- **Communication Tool:** DFDs serve as effective communication tools, allowing stakeholders to visualize and discuss the system's requirements, processes, and dataflow.



- **Analysis and Design:** DFDs facilitate system analysis and design by identifying system components, data requirements, and interactions between components.
- **Error Detection:** DFDs can help in identifying potential errors, redundancies, or inefficiencies in the system's data flow and processing logic.
- **Documentation:** DFDs provide documentation of the system's structure and behavior, serving as valuable reference material for future development, maintenance, and troubleshooting.
- **Visual Representation:** DFDs provide a clear and intuitive visual representation of the data flow within a system. This visual clarity makes it easier for stakeholders to understand the deduplication process and identify potential bottlenecks or inefficiencies.
- **Identification of Data Sources and Destinations:** DFDs help in identifying the sources of data inputs and the destinations where deduplicated data will be stored or processed. This clarity is crucial for understanding the scope of deduplication and ensuring that all relevant data sources are considered.
- **Identification of Duplicate Data Paths:** By mapping out the flow of data through the system, DFDs help in identifying duplicate data paths and potential points of duplication. This allows deduplication algorithms to be applied strategically to eliminate redundancies at key stages of the data flow.



**Fig- 4.3 : Flowchart diagram**

## **4.4 UML DIAGRAMS**

UML stands for Unified Modeling Language. UML is a standardized general- purpose modeling language in the field of object-oriented software engineering. The standard is managed, and was created by, the Object.

### **Management Group.**

The goal is for UML to become a common language for creating models of object oriented computer software. In its current form UML is comprised of **two** major components: a Meta-model and a notation. In the future, some form of method or process may also be added to; or associated with, UML.

The Unified Modeling Language is a standard language for specifying, Visualization, Constructing and documenting the artifacts of software system, as well as for business modeling and other non-software systems.

The UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems.

The UML is a very important part of developing objects oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects.

#### **4.4.1 Components of UML:**

- **Meta-model:** The meta-model defines the abstract syntax and semantics of UML elements, providing a foundation for understanding and interpreting UML diagrams. It defines the types of elements (e.g., classes, objects, relationships) and their relationships within the UML framework.
- **Notation:** UML provides a graphical notation consisting of various types of diagrams to represent different aspects of a software system or business process. Some commonly used UML diagrams include:
- **Class diagrams:** Represent the static structure of a system, including classes, attributes, methods, and their relationships.
- **Use case diagrams:** Describe the interactions between actors (users or systems) and the system under consideration.

- Illustrate the interactions between objects or components over time, showing the sequence of messages exchanged.
- Activity diagrams: Model the flow of control within a system or a specific process, depicting activities, decisions, and transitions.
- State machine diagrams: Represent the dynamic behavior of an object or system in terms of states, events, and transitions.
- Component diagrams: Show the organization and dependencies of components within a system or subsystem.

#### **4.4.2 Goals and Benefits of UML:**

- Standardization: UML aims to provide a standardized language and notation for modeling software systems, ensuring consistency and interoperability across different tools and platforms.
- Communication: UML serves as a common visual language for communication and collaboration among stakeholders involved in the software development process, including developers, architects, designers, and business analysts.
- Visualization: UML diagrams provide visual representations of system structure, behavior, and interactions, making complex concepts and relationships easier to understand and communicate.
- Abstraction: UML allows developers to abstract away implementation details and focus on high-level design concepts and requirements, facilitating better decision-making and problem-solving.

- **Documentation:** UML diagrams serve as documentation artifacts that capture and convey important aspects of a software system's design, architecture, and functionality. They provide a valuable reference for understanding and maintaining the system over time.
- **Modeling Best Practices:** UML incorporates best practices and proven techniques for modeling large and complex systems, derived from decades of experience in software engineering and system design.

## 4.5 GOALS:

The Primary goals in the design of the UML are as follows:

- **Provide users a ready-to-use, expressive visual modeling Language so that they can develop and exchange meaningful models:**

The primary purpose of UML is to provide users with a standardized and expressive visual modeling language. UML offers a comprehensive set of diagrams and notations that allow users to represent various aspects of a system or process visually. These diagrams include class diagrams, use case diagrams, sequence diagrams, activity diagrams, and more. By using UML, users can develop models that accurately capture the structure, behavior, and interactions within a system, making it easier to understand, communicate, and analyze complex systems.

- **Provide extendibility and specialization mechanisms to extend the core concepts:**

UML is designed to be extendable and customizable to meet the specific needs of different domains and industries. It provides mechanisms for extending and specializing its core concepts through mechanisms such as stereotypes, profiles, and packages. This allows users to tailor UML to their specific domain or application context, ensuring that the modeling language remains flexible and adaptable to diverse modeling requirements.

- **Be independent of particular programming languages and development**

UML is intended to be programming language and development process agnostic, meaning it can be used with any programming language and within any software development methodology. This language independence allows UML to be applied in a wide range of contexts, from object-oriented programming languages like Java and C++ to non-object-oriented languages and from traditional waterfall to agile methodologies.

- **Provide a formal basis for understanding the modeling language:**

UML is grounded in formal modeling principles and concepts, providing a solid theoretical foundation for understanding and interpreting UML diagrams. It defines precise semantics for its elements, relationships, and constructs, ensuring consistency and clarity in model interpretation. This formal basis helps users develop accurate, reliable, and consistent models that accurately reflect the underlying system or process.

- **Encourage the growth of OO tools market:**

One of the goals of UML is to foster the development and adoption of object-oriented modeling and development tools. By providing a standardized modeling language, UML encourages the creation of tools that support the creation, editing, analysis, and simulation of UML diagrams. This has led to the proliferation of UML modeling tools in the software industry, offering users a wide range of options for modeling and developing software systems.

- **Support higher level development concepts such as collaborations, frameworks, patterns and components:**

UML supports higher-level development concepts such as collaborations, frameworks, patterns, and components. It provides constructs for modeling these concepts explicitly, allowing users to capture and represent complex

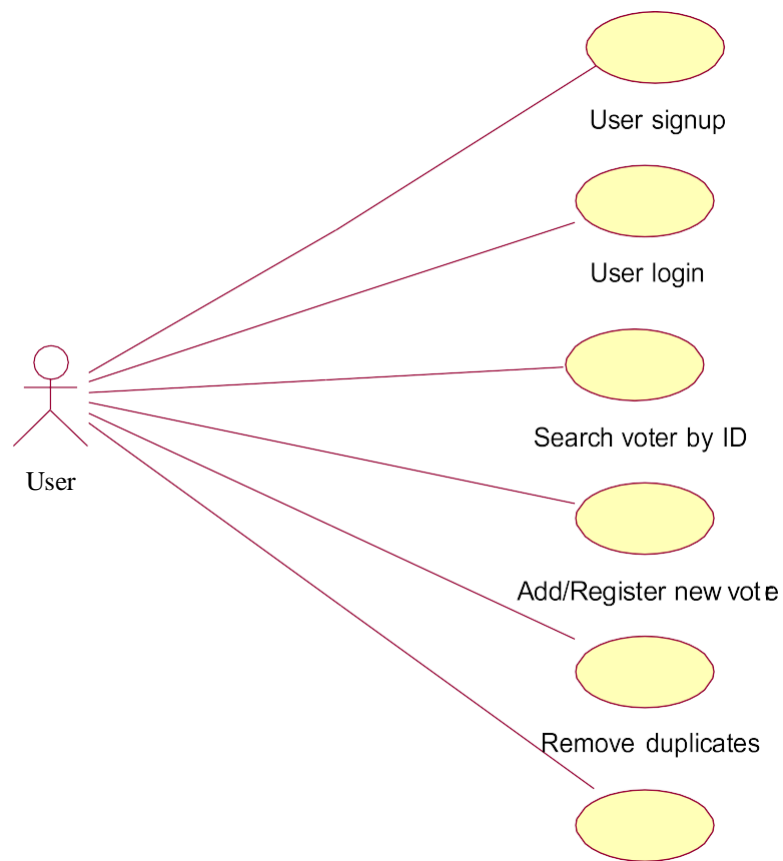
system architectures and design patterns in their models. This promotes the use of industry best practices and facilitates the reuse of proven design solutions across different projects and domains.

- **Integrate best practices:**

UML incorporates best practices and proven techniques from the field of software engineering and system design. It provides guidelines, conventions, and standards for modeling various aspects of a system, ensuring that users adhere to established modeling practices. By integrating best practices into the language itself, UML helps users develop models that are consistent, maintainable, and of high quality. and adaptable modeling language that supports a wide range of modeling requirements and promotes best practices in software engineering. UML aims to facilitate effective communication, collaboration, and understanding among stakeholders involved in the development of software systems.

#### **4.6. Use case diagram:**

A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases.

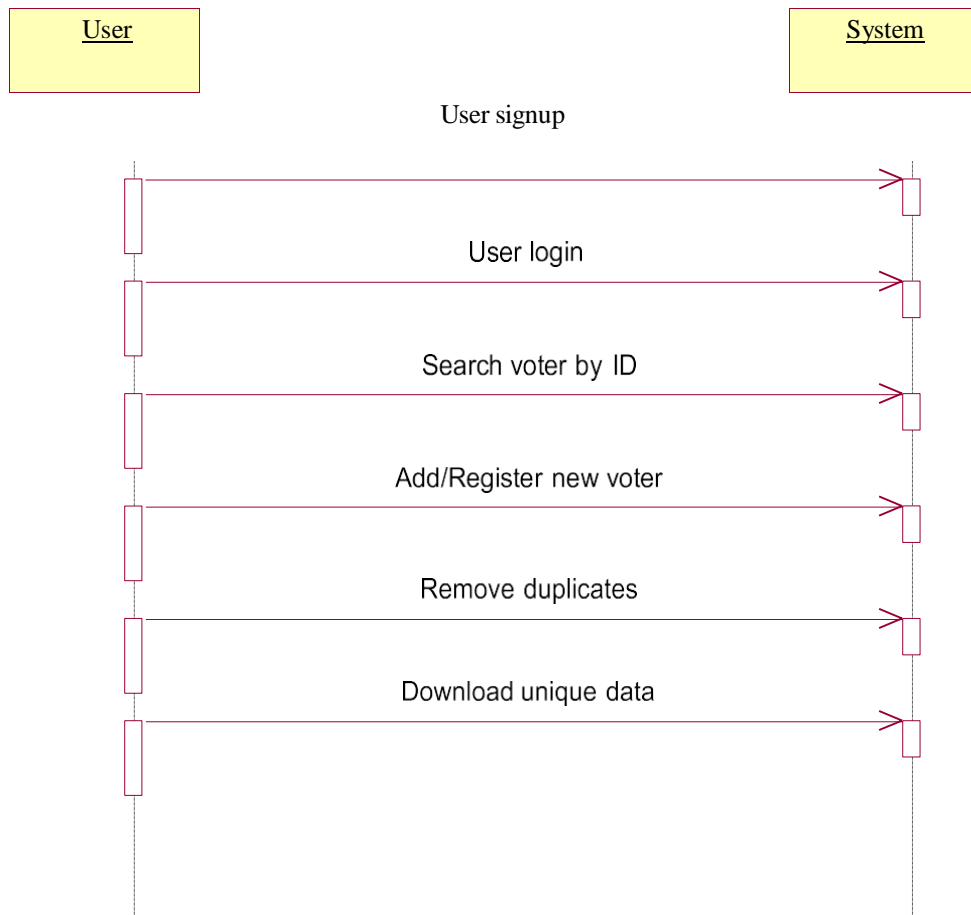


**Fig4.2.1 Use case diagram**

#### **4.7 Sequence diagram:**

A sequence diagram represents the interaction between different objects in the system. The important aspect of a sequence diagram is that it is time-ordered. This means that the exact sequence of the interactions between the objects is represented step by step. Different objects in the sequence diagram interact with each other by passing "messages".





**Fig 4.2.2 Sequence diagram**

#### **4.7 Collaboration diagram:**

A collaboration diagram groups together the interactions between different objects. The interactions are listed as numbered interactions that help to trace the sequence of the interactions. The collaboration diagram helps to identify all the possible interactions that each object has with other objects.

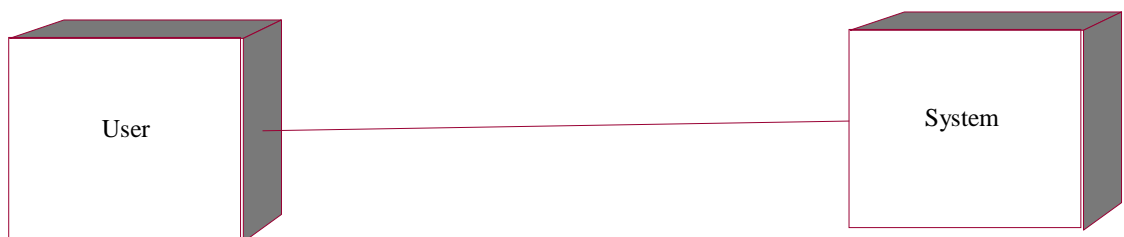
- 1: User signup
- 2: User login
- 3: Search voter by ID
- 4: Add/Register new voter
- 5: Remove duplicates
- 6: Download unique data



**Fig 4.2.3 Collaboration diagram**

## **4.8 Deployment diagram:**

The deployment diagram captures the configuration of the runtime elements of the application. This diagram is by far most useful when a system is built and ready to be deployed.



**Fig 4.2.4 Deployment diagram**

## **CHAPTER 5**

### **IMPLEMENTATION**

#### **5.1 PYTHON LANGUAGE:**

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high level built in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms and can be freely distributed.

Often, programmers fall in love with Python because of the increased productivity it provides. Since there is no compilation step, the edit-test-debug cycle is incredibly fast. Debugging Python programs is easy: a bug or bad input will never cause a segmentation fault. Instead, when the interpreter discovers an error, it raises an exception.

When the program doesn't catch the exception, the interpreter prints a stack trace. A source level debugger allows inspection of local and global variables, evaluation of arbitrary expressions, setting breakpoints, stepping through the code a line at a time, and so on. The debugger is written in Python itself, testifying to Python's introspective power. On the otherhand, often the quickest way to debug a program is to add a few print statements to the source: the fast edit-test-debug cycle makes this simple approach very effective.

Python is a dynamic, high-level, free open source, and interpreted programming language. It supports object-oriented programming as well as procedural-oriented programming. In Python, we don't need to declare the type of variable because it is a dynamically typed language. For example, `x = 10` Here, `x` can be anything such as String, int, etc.

## **Features in Python:**

There are many features in Python, some of which are discussed below as follows:

### **1. Free and Open Source**

Python language is freely available at the official website and you can download it from the given download link below click on the Download Python keyword. Download Python Since it is open-source, this means that source code is also available to the public. So you can download it, use it as well as share it

### **2. Easy to code**

Python is a high-level programming language. Python is very easy to learn the language as compared to other languages like C, C#, JavaScript, Java, etc. It is very easy to code in the Python language and anybody can learn Python basics in a few hours or days. It is also a developer-friendly language.

### **3. Easy to Read**

As you will see, learning Python is quite simple. As was already established, Python's syntax is really straightforward. The code block is defined by the indentations rather than by semicolons or brackets.

### **4. Object-Oriented Language**

One of the key features of Python is Object-Oriented programming. Python supports object-oriented language and concepts of classes, object encapsulation, etc.

### **5. GUI Programming Support**

Graphical User interfaces can be made using a module such as PyQt5, PyQt4, wxPython, or Tk in python. PyQt5 is the most popular option for creating graphical apps with Python.

### **6. High-Level Language**

Python is a high-level language. When we write programs in Python, we do not need to remember the system architecture, nor do we need to manage the memory.

### **7. Extensible feature**

Python is an Extensible language. some Python code into C or C++ language and also we can compile that code in C/C++ language.

### **8. Easy to Debug:**

Excellent information for mistake tracing. You will be able to quickly identify and correct the majority of your program's issues once you understand how to interpret Python's error traces. Simply by glancing at the code, you can determine what it is designed to perform.

### **9. Python is a Portable language**

Python language is also a portable language. For example, if Python code for windows and if we want to run this code on other platforms such as Linux, Unix, and Mac then we do not need to change it, we can run this code on any platform.

### **10. Python is an integrated language**

Python is also an integrated language because easily integrate Python with other languages like C, C++, etc.

### **11. Interpreted Language**

Python is an Interpreted Language because Python code is executed line by line at a time. like other languages C, C++, Java, etc. there is no need to compile Python code this makes it easier to debug our code. The source code of Python is converted into an immediate form called byte code.

### **12. Large Standard Library**

Python has a large standard library that provides a rich set of modules and functions so you do not have to write your own code for every single thing. There are many libraries present in Python such as regular expressions, unit-testing, web browsers, etc.

### **13. Dynamically Typed Language**

Python is a dynamically-typed language. That means the type (for example- int, double, long, etc.) for a variable is decided at run time not in advance because of this feature we don't need to specify the type of variable.

#### **14. Frontend and backend development**

With a new project py script, you can run and write Python codes in HTML with the help of some simple tags

<py-script>, <py-env>, etc. This will help you do frontend development work in Python like JavaScript

Backend is the strong forte of Python it's extensively used for this work cause of its frameworks like Django and Flask.

#### **15. Allocating Memory Dynamically**

In Python, the variable data type does not need to be specified. The memory is automatically allocated to a variable at runtime when it is given a value. Developers do not need to write `int y = 18` if the integer value 15 is set to y. You may just type `y=18`.

### **DJANGO:**

Django is a high-level Python web framework that encourages rapid development and clean, pragmatic design. It is widely used for building web applications due to its scalability, flexibility, and extensive feature set. Here are some key features of Django:

**Model-View-Controller (MVC) Architecture:** Django follows the Model-View-Template (MVT) pattern, which is similar to the Model-View-Controller (MVC) pattern. It separates the data model, user interface, and application logic, promoting code organization and maintainability.

**ORM (Object-Relational Mapping):** Django includes a powerful ORM that abstracts away the details of database management, allowing developers to interact with the database using Python objects. It supports various database backends, including PostgreSQL, MySQL, SQLite, and Oracle.

**Admin Interface:** Django provides an automatically generated admin interface for managing site content and database records. Developers can define models and register them with the admin site to enable CRUD (Create, Read, Update, Delete) operations without writing additional code.

**URL Routing:** Django uses a URL routing mechanism to map URLs to views, allowing developers to define URL patterns and associate them with corresponding view functions or class-based views. This makes it easy to create clean and expressive URL structures.

**Template Engine:** Django includes a built-in template engine that enables developers to create HTML templates with dynamic content using template tags and filters. Templates support inheritance, inclusion, and other powerful features for building reusable and maintainable UI components.

**Form Handling:** Django provides a form handling library that simplifies the process of validating and processing user input. Developers can define forms using Python classes and leverage built-in form widgets, validators, and error handling mechanisms.

**Authentication and Authorization:** Django includes robust authentication and authorization mechanisms for user authentication, session management, and access control. It supports user authentication using username/password, social authentication, and custom authentication backends.

**Internationalization and Localization:** Django supports internationalization (i18n) and localization (l10n) features for building multilingual web applications. It provides tools for translating text strings, formatting dates and numbers, and serving content in different languages based on user preferences.

**Extensibility and Reusability:** Django's modular architecture and pluggable app system promote code reusability and extensibility. Developers can create reusable components (apps) and share them with the Django community through the Python Package Index (PyPI), fostering a rich ecosystem of third-party packages and libraries.

## 5.2 LIBRARIES/ PACKAGES:

### **Tensor flow**

Tensor Flow is a free and open-source software library for dataflow and differentiable programming across a range of tasks. It is a symbolic math library, and is also used for machine learning applications such as neural networks. It is used for both research and production at Google.

TensorFlow was developed by the Google Brain team for internal Google use. It was released under the Apache 2.0 open-source license on November 9, 2015.

### **Numpy**

Numpy is a general-purpose array-processing package. It provides a high-performance multidimensional array object, and tools for working with these arrays. It is the fundamental package for scientific computing with Python. It contains various features including these important ones:

- A powerful N-dimensional array object
- Sophisticated (broadcasting) functions
- Tools for integrating C/C++ and Fortran code
- Useful linear algebra, Fourier transform, and random number capabilities

NumPy allows for vectorized operations on arrays, which means that operations are applied element-wise without the need for explicit looping, leading to faster and more concise code.

NumPy provides a wide range of mathematical functions that operate element-wise on arrays, including basic arithmetic operations, trigonometric functions, exponential and logarithmic functions, and more.

NumPy's broadcasting rules enable operations between arrays of different shapes and sizes, automatically aligning dimensions to perform element-wise operations efficiently.

It offers powerful indexing and slicing capabilities for accessing and manipulating data within arrays, including advanced indexing techniques like boolean indexing, integer array indexing, and fancy indexing.



Besides its obvious scientific uses, Numpy can also be used as an efficient multi- dimensional container of generic data. Arbitrary data-types can be defined using Numpy which allows Numpy to seamlessly and speedily integrate with a wide variety of databases.

### **Pandas**

Pandas is an open-source Python Library providing high-performance data manipulation and analysis tool using its powerful data structures. Python was majorly used for data munging and preparation. It had very little contribution towards data analysis. Pandas solved this problem. Using Pandas, we can accomplish five typical steps in the processing and analysis of data, regardless of the origin of data load, prepare, manipulate, model, and analyze. Python with Pandas is used in a wide range of fields including academic and commercial domains including finance, economics, Statistics, analytics, etc.

### **Matplotlib**

Matplotlib is a Python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms. Matplotlib can be used in Python scripts, the Python and IPython shells, the Jupyter Notebook, web application servers, and four graphical user interface toolkits. Matplotlib tries to make easy things easy and hard things possible. You can generate plots, histograms, power spectra, bar charts, error charts, scatter plots, etc., with just a few lines of code. For examples, see the sample plots and thumbnail gallery. For simple plotting the pyplot module provides a MATLAB-like interface, particularly when combined with IPython. For the power user, you have full control of line styles, font properties, axes properties, etc, via an object- oriented interface or via a set of functions familiar to MATLAB users.

### **Scikit – learn**

Scikit-learn provide a range of supervised and unsupervised learning algorithms via a consistent interface in Python. It is licensed under a permissive simplified BSD license and is distributed under many Linux distributions, encouraging academic and commercial use.

### 5.3 FRONTEND:

The frontend, also known as client-side development, refers to the portion of a website or application that users directly interact with. It encompasses the visual elements, layout, and interactivity that determine the user experience (UX).

- **Focuses on User Interface (UI) and User Experience (UX):** Front-end development deals with everything users see and interact with on a website or web application. This includes layout, visuals, responsiveness, and overall user experience.
- **Core Technologies:** The essential tools for front-end development are HTML (structure and content), CSS (styling and visual design), and JavaScript (interactivity and dynamic behavior).
- **Responsiveness:** Websites need to adapt to different screen sizes and devices (desktops, mobiles, tablets). Front-end developers ensure a seamless experience across various platforms using responsive design techniques.
- **Accessibility:** Making websites usable for everyone, including people with disabilities, is crucial. Front-end developers incorporate accessibility best practices to cater to diverse users.
- **Performance Optimization:** Fast loading times are essential for a positive user experience. Front-end developers optimize code, images, and other resources for speed and efficiency.
- **Cross-Browser Compatibility:** Ensuring a website functions consistently across different web browsers is vital. Front-end developers test and refine code to achieve compatibility.
- **JavaScript Frameworks and Libraries:** Beyond core languages, frameworks like React, Angular, and Vue.js offer pre-built components and functionalities, streamlining development.

## 5.4 DATABASE:

MySQL can be utilized as the relational database management system (RDBMS) to store metadata associated with the deduplicated data. Here's a detailed explanation of how MySQL can be used in the project:

- **Database Schema Design:**

Define a database schema to represent metadata attributes related to the deduplicated data. This schema should include tables to store information such as block identifiers, checksums, timestamps, data sources, and deduplication status. Design normalized tables with appropriate data types, constraints, and relationships to ensure data integrity and efficiency.

- **Data Storage:**

Insert records into the MySQL tables to store metadata about the deduplicated data blocks. Use INSERT statements to add data to the tables, ensuring that each record includes all relevant metadata attributes. Implement batch processing techniques if necessary to optimize data insertion performance.

- **Data Retrieval:**

Develop SQL queries to retrieve metadata from the MySQL database based on various criteria such as block identifiers, timestamps, or deduplication status. Use SELECT statements with appropriate WHERE clauses to filter and fetch the required metadata records. Optimize query performance by creating indexes on columns frequently used in search and retrieval operations.

- **Data Maintenance:**

Implement mechanisms to update, delete, or modify metadata records in the MySQL database as needed. Use UPDATE and DELETE statements to modify or remove existing records based on specific criteria. Implement transaction management to ensure data consistency and atomicity during data modification.

- **Security:**

Implement security measures to protect MySQL database access and data confidentiality. Configure authentication and authorization settings to restrict access to authorized users and roles. Enforce encryption for data in transit and at rest to safeguard sensitive information stored in the database.

## **5.5 DJANGO FRAMEWORK**

Django is a high-level Python web framework that encourages rapid development and clean, pragmatic design. It's built to help developers create web applications quickly, with a focus on simplicity, flexibility, and scalability.

Here's a breakdown of some key aspects of Django:

### **1. MVC Architecture:**

Django follows the Model-View-Controller (MVC) architectural pattern, although it refers to it as MTV (Model-Template-View). This separation of concerns allows for easier maintenance and scalability of web applications.

### **2. ORM (Object-Relational Mapping):**

Django provides an ORM that enables developers to interact with the database using Python objects. This abstraction makes it easier to work with databases without needing to write SQL queries directly, thus promoting code readability and maintainability.

### **3. Admin Interface:**

Django comes with a built-in admin interface that can be used to manage site content without much effort. Developers can quickly create, update, and delete records in the database through this interface, which is highly customizable and extensible.

### **4. URL Routing:**

Django uses a URL routing system that maps URLs to Python code. This makes it easy to create clean and readable URLs for different views and functionalities within the web application.

### **5. Template System:**

Django provides a powerful template system that allows developers to build HTML templates with Python-like syntax. This makes it easy to create dynamic web pages by combining HTML with data from the backend.

### **6. Security Features:**

Django comes with built-in security features to help developers build secure web applications. This includes protection against common web vulnerabilities like SQL injection, cross-site scripting (XSS), and cross-site request forgery (CSRF).

### **7. Authentication and Authorization:**

Django provides robust authentication and authorization mechanisms out of the box. Developers can easily integrate user authentication, permissions, and user management into their applications.

### **8. Form Handling:**

Django simplifies form handling by providing a form library that helps developers create and process HTML forms easily. It includes features like form validation, error handling, and CSRF protection.

### **9. Middleware:**

Django uses middleware components to process requests and responses. Middleware can perform tasks such as authentication, logging, or modifying HTTP headers before passing requests to the views.

### **10. Internationalization and Localization:**

Django supports internationalization (i18n) and localization (l10n) features, making it easy to build multilingual web applications that can be adapted to different languages and regions.

## 5.4 SETTING THE ENVIRONMENT:

Before we dive into the world of deploying de-duplication algorithm with FastAPI, we need to ensure our development environment is properly set up. This section will guide you through the process step by step.

### **Install Python:**

First and foremost, ensure you have Python installed on your system. You can download the latest version of Python from the official website (<https://www.python.org/downloads/>) or use a package manager like Anaconda.

To check if Python is already installed, open your terminal or command prompt and run:

```
python -version
```

To keep our project's dependencies isolated, it's a good practice to create a virtual environment. This way, you can avoid conflicts between different projects.

Let's create a virtual environment using Python's built-in venv module. Open your terminal and navigate to your project's root directory.

Run the following commands:

```
Python -m venv myenv
```

```
myenv\Scripts\activate
```

```
source myenv/bin/activate
```

You should see your terminal prompt change to indicate that the virtual environment is active.

### **Install Dependencies**

Now that you're working within your virtual environment, it's time to install the necessary dependencies. These include required libraries, and any additional packages you may need for the project.

## 5.5 DE-DUPLICATION ALOGORITHM:

Data deduplication is a specific form of compression where redundant data is eliminated, typically to improve storage utilization. In the deduplication process, duplicate data is deleted, leaving only one copy of the data to be stored. However, indexing of all data is still retained should that data ever be required. Deduplication is able to reduce the required storage capacity since only the unique data is stored.

### Methods For De-Duplication Algorithm

- File-level Deduplication
- Block-level Deduplication

**1. File-level deduplication** watches for multiple copies of the same file, stores the first copy, and then just links the other references to the first file. Only one copy gets stored on the disk/tape archive. Ultimately, the space you save on disk relates to how many copies of the file there were in the file system.

Example: Let's assume a company having a 1000 employee share a common file say "data.txt" which is 10MB in Size. Each employee do the same changes and save the exact similar 1000 copies of file on server. so estimated storage require to save a file on server side is 10 GB.

**2. Block-level Deduplication**, sometimes called variable block-level deduplication, looks at the data block itself to see if another copy of this block already exists. If so, the second (and subsequent) copies are not stored on the disk/tape, but a link/pointer is created to point to the original copy.

```

def update_hash_dict(self):
    # Calculate sets of current and cached filenames
    current_files = set(self.image_filenames)
    cache_files = self.hash_dict.keys()
    # Identify lost and new files
    lost_set = cache_files - current_files
    target_files = list(current_files - cache_files)
    # Update hash dictionary for new files if len(lost_set) + len(target_files) > 0:
    try:
        # Handle spinner display if len(self.hash_dict) == 0:
        Spinner = Spinner(prefix="Calculating image hashes (hash-bits={ } num-
        proc={ })...".format(self.hash_bits, self.num_proc))
    else:
        Spinner = Spinner(prefix="Updating image hashes (hash-bits={ } num-
        proc={ })...".format(self.hash_bits, self.num_proc))
    spinner.start()
    # Perform parallel processing to compute hashes for new files if six.PY2: from
    pathos.multiprocessing import ProcessPool as Pool elif six.PY3: from multiprocessing
    import Pool
    pool = Pool(self.num_proc)
    hashes = pool.map(self.gen_hash, target_files)
    # Update hash dictionary with new hashes
    for filename, hash_value in zip(target_files, hashes):
        self.hash_dict[filename] = hash_value
    spinner.stop()
    except KeyboardInterrupt:
        pool.terminate()
        pool.join()
        spinner.stop()
        sys.exit(1)
    return True
else:
    return False

```

The deduplication algorithm in the provided code is used to ensure that duplicate text are not processed redundantly, thereby optimizing the performance by avoiding unnecessary computations. It involves comparing the list of repeated text with the existing hashdictionary to identify repeated text that are either new or have been removed since the last hashing operation.



Let's break down how this algorithm works

- The algorithm begins by obtaining two sets: `current_files` representing the current list of text filenames and `cache_files` representing the filenames already present in the hash dictionary.
- It then calculates the set difference between `cache_files` and `current_files` to identify repeated text that were present in the previous hash dictionary but are no longer present in the current list (`lost_set`).
- Similarly, it calculates the set difference between `current_files` and `cache_files` to identify new text that need to be hashed (`target_files`).
- The algorithm then proceeds to update the hash dictionary only for the new text (`target_files`), removing hashes for repeated text that are no longer present (`lost_set`).
- The deduplication algorithm is primarily used in the `update_hash_dict()` method within the `HashCache` class.
- This method is called whenever there is a need to synchronize the hash dictionary with the current state of the repeated texts.
- By deduplicating the list of repeated text and updating the hash dictionary accordingly, it ensures that only new text are hashed, and the dictionary remains up to date with the file system.

## **5.6 MODULES:**

We are using 'DataVoter.xlsx' as the main database file and new voter will be added to it and duplicates will be removed from that file. We have added all given options like

### **1. User Module:**

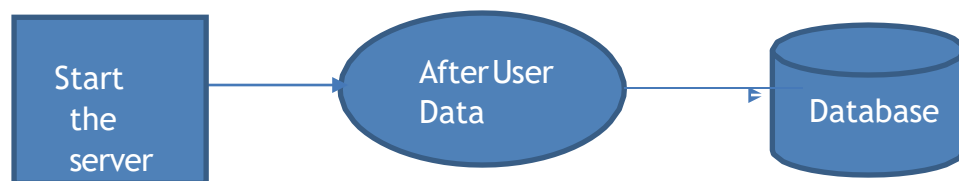
In this module, Users are having authentication and security to access the detail which is presented in the ontology system. Before accessing or searching the details user should have the account in that otherwise they should register first. At the very least, you need to provide an email address, username, password, display name, and whatever profile fields you have set to required. The display name is what will be used when the system needs to display the proper name of user.



**Fig 5.8.1 User Module**

## **2. Server Start Up and Upload :**

The user can start up the server after cloud environment is opened. Then the user can upload the file to the cloud.



**Fig 5.8.2 Server Module**

## **3. Secure De-Duplication System:**

To support authorized de duplication the tag of a file  $F$  will be determined by the file  $F$  and the privilege. To show the difference with traditional notation of tag, we call it file token instead. To support authorized access a secret key  $KP$  will be bounded with a privilege  $p$  to generate a file Token. De duplication exploits identical content, while encryption attempts to make all content appear random; the same content encrypted with two different keys results in very different cipher text. Thus, combining the space efficiency of de duplication with the secrecy aspects of encryption is problematic.

## **4. Download File:**

After the cloud storage, the user can download the file based on key or token. Once the key request was received, the sender can send the key or he can decline it. With this key and request id which was generated at the time of sending key request the receiver can decrypt the message.

## 5.7 PESUDO CODE:

### De-Duplication Alogorithm:

```
class HashCache:
    constructor(args, image_filenames, hash_method, hash_size, num_proc, load_path=None):
        self.args = args
        self.image_filenames = image_filenames
        self.hashfunc = self.gen_hashfunc(hash_method)
        self.hash_size = hash_size
        self.hash_bits = hash_size ** 2
        self.num_proc = num_proc
        self.hash_dict = { }

    method len():
        return length of cache

    method hshs():
        return list of hash values in hash_dict

    method filenames():
        return list of filenames in hash_dict

    method gen_hash(img):try:
        Open img as i
        Calculate hash value hsh using hashfunc
        Convert hsh to numpy array
    except:
        Set hsh to an array of 2's
    return hsh

    method make_hash_list():
        Set num_proc if None to cpu_count() - 1
```

Try:

Start spinner

Create a pool of num\_proc processes

Calculate hashes for image\_filenames using pool.map  
Stop spinner

except KeyboardInterrupt:

Terminate pool and exit  
return

method update\_hash\_dict():

Set num\_proc if None to cpu\_count() - 1  
Get current\_files and cache\_files  
Calculate lost\_set and target\_files

If lost\_set or target\_files are not empty:  
Start spinner

Remove files in lost\_set from hash\_dict  
Create a pool of num\_proc processes

Calculate hashes for target\_files using pool.map  
Update hash\_dict with new hashes

Stop spinner

Return True if update occurred else False

method phash\_org(image, hash\_size=8, highfreq\_factor=4):

Calculate hash value of the image using DCT

Return the hash value

method gen\_hashfunc(hash\_method):

Return appropriate hash function based on hash\_method

method load\_hash\_dict(load\_path, use\_cache, target\_dir):

If load\_path exists and use\_cache is True:

Log loading of hash cache  
Start

spinner

```
Load hash cache using joblib
Stop
spinner
Check if update occurred
Return True if cache loaded and not updated else False
Else:
Initialize hash_dict to empty
Update
hash_dict
Return False
method dump_hash_dict(dump_path, use_cache):
If
use_cache is True:
Dump hash_dict to dump_path using joblib
Log
dumping of hash cache
Return True
Else:
Return False
```

## 5.8 CODE:

```
from logging import getLogger, StreamHandler, DEBUG logger = getLogger( name )handler
= StreamHandler() # handler.setLevel(DEBUG) # logger.setLevel(DEBUG)
logger.addHandler(handler) logger.propagate = F

from multiprocessing import cpu_count from pathlib import Path from PIL import Image,
ImageFile from tqdm

import tqdm

import imagehash import joblib import sys import six import numpy import scipy from
common.spinner import Spinner ImageFile.LOAD_TRUNCATED_IMAGES =True class
HashCache:

def init (self, args, image_filenames, hash_method, hash_size, num_proc,load_path=None):

self.args = args

self.image_filenames = image_filenames self.hashfunc =
self.gen_hashfunc(hash_method) self.hash_size = hash_size

self.hash_bits = hash_size ** 2 self.num_proc = num_proc self.hash_dict = { } deflen (self):
return len(self.cache)

def hshs(self):

return list(self.hash_dict.values())def

filenames(self):

return list(self.hash_dict.keys())def

gen_hash(self, img): try:

with Image.open(img) as i:

hsh = self.hashfunc(i, hash_size=self.hash_size)

hsh = numpy.array([ 1 if b else 0 for b in hsh.hash.reshape((self.hash_bits))]) except:hsh =
numpy.array([2] * self.hash_bits) return hsh

def make_hash_list(self):
```

```

if self.num_proc is None: self.num_proc = cpu_count() - 1
try:
    spinner = Spinner(prefix="Calculating image hashes (hash-bits={ } num-
proc={ })..." .format(self.hash_bits, self.num_proc))

    spinner.start()
    if six.PY2:
        from pathos.multiprocessing import ProcessPool as Pool
    elif six.PY3:
        from multiprocessing
import Pool
pool

= Pool(self.num_proc)

self.cache = pool.map(self.gen_hash,self.image_filenames)
spinner.stop()
except
KeyboardInterrupt: pool.terminate() pool.join()

spinner.stop()
sys.exit(1)
def

update_hash_dict(self):

if self.num_proc is None: self.num_proc = cpu_count() - 1
# check

current_hash_dict

current_files = set(self.image_filenames)
cache_files = self.hash_dict.keys()
lost_set

= cache_files - current_files

target_files = list(current_files - cache_files)
if

len(lost_set) + len(target_files) > 0:

try:

if len(self.hash_dict) == 0:

    spinner = Spinner(prefix="Calculating image hashes (hash-bits={ } num
proc={ })..." .format(self.hash_bits, self.num_proc))

else:

    spinner = Spinner(prefix="Updating image hashes (hash-bits={ } num-
proc={ })..." .format(self.hash_bits, self.num_proc))

    spinner.start()

```

```

# del lost_set from hash_dict for f in lost_set: del self.hash_dict[f] if six.PY2:from
pathos.multiprocessing import ProcessPool as Pool elif six.PY3:

from multiprocessing import Pool pool = Pool(self.num_proc) hashes =
pool.map(self.gen_hash, target_files) for filename, hash_value in zip(target_files,hashes):
self.hash_dict[filename] = hash_value spinner.stop() else:

except KeyboardInterrupt: pool.terminate() pool.join() spinner.stop() sys.exit(1) return True
return False

def phash_org(self, image, hash_size=8, highfreq_factor=4): if hash_size < 2: raise
ValueError("Hash size must be greater than or equal to 2")

import scipy.fftpack

img_size = hash_size * highfreq_factor

image = image.convert("L").resize((img_size, img_size), Image.ANTIALIAS) pixels
= numpy.asarray(image) dct = scipy.fftpack.dct(scipy.fftpack.dct(pixels, axis=0),axis=1)

# using only the 8x8 DCT low-frequency values and excluding the first term since theDC
coefficient

# can be significantly different from the other valuesand will throw off the average.
dctlowfreq = dct[1:hash_size+1, 1:hash_size+1]

med = numpy.median(dctlowfreq) diff = dctlowfreq > med return
imagehash.ImageHash(diff)

def gen_hashfunc(self, hash_method): if hash_method == 'ahash':

hashfunc = imagehash.average_hash elif hash_method == 'phash':

hashfunc = imagehash.phash

elif hash_method == 'dhash': hashfunc = imagehash.dhash elif hash_method == 'whash':
hashfunc =

Imagehash.whash elif hash_method == 'phash_org': hashfunc = self.phash_org returnhashfunc

```



```

def load_hash_dict(self, load_path, use_cache, target_dir): if load_path and
Path(load_path).exists() and use_cache:

logger.debug("Load hash cache: {}".format(load_path)) spinner =Spinner(prefix="Loading
hash cache...") spinner.start()

self.hash_dict = joblib.load(load_path) spinner.stop() is_update =self.update_hash_dict()
return not is_update else:

self.hash_dict = { self.update_hash_dict()

return False

def dump_hash_dict(self, dump_path, use_cache): if use_cache:

joblib.dump(self.hash_dict, dump_path, protocol=2, compress=True)logger.debug("Dump
hash cache:

{}".format(dump_path)) return True else:return

False

```

## **5.9 SYSTEM TESTING:**

System testing, also referred to as system-level tests or system-integration testing, is the process in which a quality assurance (QA) team evaluates how the various components of an application interact together in the full, integrated system or application. System testing verifies that an application performs tasks as designed. This step, a kind of black box testing, focuses on the functionality of an application. System testing, for example, might check that every kind of user input produces the intended output across the application.

### **Phases of system testing:**

A video tutorial about this test level. System testing examines every component of an application to make sure that they work as a complete and unified whole. A QA team typically conducts system testing after it checks individual modules with functional or user-story testing and then each component through integration testing.

If a software build achieves the desired results in system testing, it gets a final check via acceptance testing before it goes to production, where users consume the software. An app-dev team logs all defects, and establishes what kinds and amount of defects are tolerable.

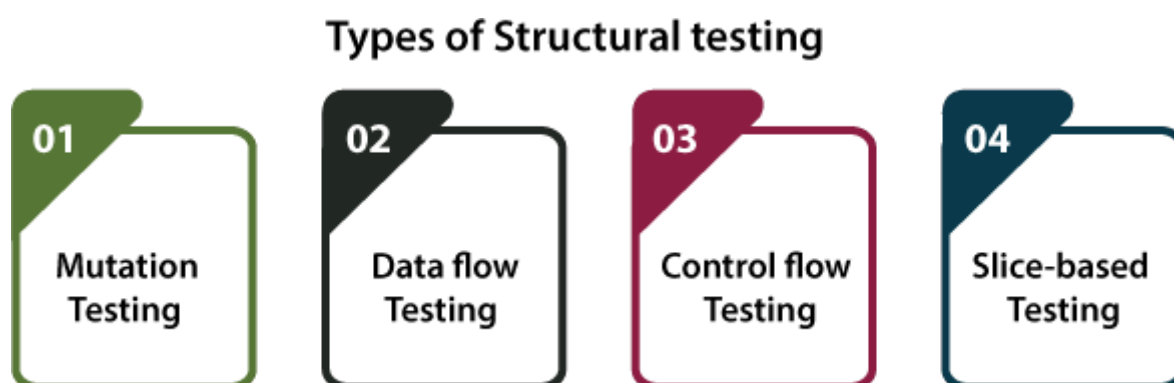
### **5.11.1 Software Testing Strategies:**

Optimization of the approach to testing in software engineering is the best way to make it effective. A software testing strategy defines what, when, and how to do whatever is necessary to make an end-product of high quality. Usually, the following software testing strategies and their combinations are used to achieve this major objective.

### **5.11.2 Structural Testing:**

It is not possible to effectively test software without running it. Structural testing, also known as white-box testing, is required to detect and fix bugs and errors emerging during the pre-production stage of the software development process. At this stage, unit testing based on the software structure is performed using regression testing. In most cases, it is an automated process working within the test automation framework to speed up

the development process at this stage. Developers and QA engineers have full access to the software's structure and data flows (data flows testing), so they could track any changes (mutation testing) in the system's behavior by comparing the tests' outcomes with the results of previous iterations (control flow testing).



**Fig.5.10.2 Structural testing**

**Purpose of Structural Testing:**

5.11.2.1 Detecting Bugs and Errors: Structural testing aims to uncover defects, bugs, and errors in the codebase by examining its internal logic, paths, and data flows.

5.11.2.2 Ensuring Code Coverage: It helps in achieving adequate code coverage by verifying that all parts of the code have been executed and tested.

5.11.2.3 Validating Software Architecture: Structural testing ensures that the software architecture and design adhere to the specified requirements and standards.

5.11.2.4 Structural testing can be used to ensure that the software adheres to coding standards, guidelines, and regulatory requirements. By verifying that the code follows established best practices, organizations can mitigate risks and ensure compliance with industry standards.

**Key Characteristics:**

- **White-Box Approach:** Unlike black-box testing, which focuses on the external behavior of the software, white-box testing involves examining the internal structure of the code, including control flow, data flow, and conditional statements.
- **Access to Source Code:** Developers and QA engineers have access to the source code, allowing them to understand the implementation details and design decisions.
- **Unit Testing:** Structural testing often involves unit testing, where individual components or units of the software are tested in isolation. Unit tests verify the correctness of small units of code, such as functions or methods.
- **Regression Testing:** Structural testing often includes regression testing, which ensures that recent code changes do not introduce new defects or regressions into the software.

**Common Techniques and Tools:**

- **Code Review:** Reviewing the source code for potential issues, such as coding standards violations, logic errors, or security vulnerabilities.
- **Static Analysis:** Using static analysis tools to analyze the source code without executing it. These tools can identify potential bugs, code smells, and security vulnerabilities.
- **Dynamic Analysis:** Executing the code and observing its behavior during runtime. Dynamic analysis tools can detect runtime errors, memory leaks, and performance issues.
- **Coverage Analysis:** Measuring code coverage to ensure that all parts of the code have been exercised by tests. Coverage analysis tools track which lines, branches, and paths of the code have been executed during testing.
- **Automated Testing:** Writing automated tests, such as unit tests, integration tests, and end-to-end tests, to verify the behavior and functionality of the software. Test automation frameworks facilitate the creation and execution of automated tests.

### **Advantages of Structural Testing:**

- **Early Detection of Defects:** Structural testing allows defects to be identified and fixed early in the development process, reducing the cost and effort of addressing issues later.
- **Improved Code Quality:** By thoroughly examining the internal structure of the code, structural testing helps improve code quality, reliability, and maintainability.
- **Increased Confidence:** Structural testing provides developers and stakeholders with confidence in the correctness and robustness of the software.
- **Facilitates Continuous Integration:** Automated structural testing can be integrated into continuous integration (CI) pipelines, enabling rapid feedback on code changes and ensuring the stability of the software.

### **5.11.3 Behavioral Testing:**

The final stage of testing focuses on the software's reactions to various activities rather than on the mechanisms behind these reactions. In other words, behavioral testing, also known as black-box testing, presupposes running numerous tests, mostly manual, to see the product from the user's point of view. QA engineers usually have some specific information about a business or other purposes of the software ('the black box') to run usability tests, for example, and react to bugs as regular users of the product will do. Behavioral testing also may include automation (regression tests) to eliminate human error if repetitive activities are required. For example, you may need to fill 100 registration forms on the website to see how the product copes with such an activity, so the automation of this test is preferable.

Behavioral testing, also known as black-box testing, is a software testing technique that focuses on evaluating the behavior and functionality of a software application without delving into its internal structure, implementation details, or code logic. Instead, behavioral testing examines how the software responds to various inputs and stimuli, simulating real-world user interactions and scenarios. This approach views the software as a "black box," where the internal workings are not visible, and only the inputs and outputs are considered.

**purpose of Behavioral Testing:**

5.11.3.1 Assessing User Experience: Behavioral testing evaluates the software from the perspective of end- users, assessing factors such as ease of use, intuitiveness, and overall user experience.

5.11.3.2 Validating Requirements: It ensures that the software meets the specified requirements and performs as expected in real-world scenarios, regardless of its internal implementation.

5.11.3.3 Identifying Defects: Behavioral testing helps identify defects, errors, and inconsistencies in the software's behavior, such as incorrect outputs, unexpected responses, or usability issues.

5.11.3.4 Ensuring Functional Compliance: It verifies that the software functions correctly according to functional specifications, business rules, and regulatory requirements.

**Key Characteristics:**

5.11.3.5 Black-Box Approach: Unlike white-box testing, which examines the internal structure of the code, black- box testing focuses solely on the external behavior and functionality of the software.

5.11.3.6 Limited Knowledge of Internals: QA engineers performing behavioral testing typically have limited or no knowledge of the software's internal workings, mimicking the perspective of end-users.

5.11.3.7 Real-World Scenarios: Behavioral tests simulate real-world usage scenarios, including user interactions, data inputs, and expected outputs, to validate the software's behavior under normal and edge cases.

5.11.3.8 Manual and Automated Testing: While behavioral testing often involves manual testing conducted by QA engineers, automation tools and scripts may also be employed, particularly for repetitive or regression testing tasks.

## **Common Techniques and Tools:**

**Functional Testing:** Testing the software's functionality against specified requirements, user stories, and use cases to ensure that it behaves as expected.

**Usability Testing:** Evaluating the software's user interface (UI), navigation flows, and overall usability to assess its ease of use and user satisfaction.  
**Exploratory Testing:** Ad-hoc testing conducted by QA engineers to explore the software's features, uncover defects, and validate user workflows.

**User Acceptance Testing (UAT):** Involving end-users or stakeholders to validate the software's functionality, usability, and compliance with business

requirements.  
**Regression Testing:** Re-running previously executed tests to ensure that recent code changes or updates have not introduced new defects or regressions. **Test**

**Automation Tools:** Using automation tools and frameworks to automate repetitive tests and scenarios, improving efficiency and consistency in testing

**Boundary Value Analysis (BVA):** BVA is a technique used to test the behavior of the software at the boundaries of input ranges. It helps identify defects related to boundary conditions. Tools like QTest and QMetry support boundary value

analysis.  
**Equivalence Partitioning:** This technique divides the input domain into equivalence classes to reduce the number of test cases while ensuring adequate test coverage. Tools like TestComplete and Ranorex assist in equivalence partitioning.

**State Transition Testing:** State transition testing focuses on testing the behavior of the software as it transitions between different states or modes. Tools like SpecFlow and Robot Framework can be used for state transition testing.

**Load Testing:** Load testing is performed to evaluate the software's behavior under normal and peak load conditions. Tools like Apache JMeter and LoadRunner are commonly used for load testing.

### **Advantages of Behavioral Testing:**

- **User-Centric Approach:** Behavioral testing focuses on the end-user perspective, ensuring that the software meets user expectations and delivers a satisfactory user experience.
- **Detecting User-Visible Defects:** By simulating real-world interactions, behavioral testing helps identify defects and issues that impact users directly, such as UI glitches, navigation problems, or incorrect outputs.
- **Validation of Business Requirements:** It validates that the software meets business requirements, functional specifications, and regulatory standards, providing confidence to stakeholders and customers.
- **Flexibility and Adaptability:** Behavioral testing allows QA engineers to adapt testing strategies and scenarios based on evolving user needs, feedback, and changing requirements.

### **5.11.4 Black Box Testing:**

- Black box testing is a valuable software testing technique that evaluates the functionality and behavior of a software application from an external perspective.
- By focusing on inputs, outputs, and user interactions, black box testing helps ensure that the software meets requirements, functions correctly, and delivers a satisfactory.
- Black box testing employs techniques like decision table testing to test combinations of inputs and state transition testing to examine the behavior of the system as it transitions between different states.
- Black box testing often employs equivalence partitioning to group input values into equivalence classes, simplifying the creation of test cases while ensuring adequate coverage.
- Testers approach black box testing from the perspective of an end user, focusing on inputs, outputs, and system behavior.



## **Key Aspects of Black Box Testing:**

**External Perspective:** Testers do not have access to the source code or internal workings of the software being tested. They interact with the software solely through its user interface, APIs, or other external interfaces.

**Testing Scenarios:** Black box testing encompasses various testing scenarios, including functional testing, usability testing, integration testing, and system testing. Each scenario focuses on different aspects of the software's behavior.

**Limited Knowledge:** Testers performing black box testing have limited knowledge of the software's internal implementation. They rely on specifications, requirements documents, user stories, and other external sources of information to guide their testing efforts.

**Input-Output Analysis:** Testers provide inputs to the software and analyze its outputs to verify whether it behaves as expected. They compare the actual outputs against expected outcomes based on predefined test cases or user expectations.

**Equivalence Partitioning:** This technique divides the input domain of the software into equivalence classes, where inputs within the same class should produce similar behavior from the software. Test cases are then designed to cover each equivalence class.

**Boundary Value Analysis:** Test cases are designed to test the boundary conditions of input ranges, as these boundaries often represent areas where errors are more likely to occur. Testers focus on inputs at the edges of valid ranges and just outside them.

**State Transition Testing:** Particularly relevant for systems with complex state-based behavior, this technique focuses on testing transitions between different states of the software, ensuring that state changes occur correctly and consistently.

**Ad-Hoc Testing:** Testers may also perform ad-hoc testing, where they explore the software's functionality and behavior in an unstructured manner, uncovering defects and issues through exploration and experimentation.

## **Techniques and Tools:**

**Manual Testing:** Testers manually execute test cases, interact with the software's user interface, and observe its behavior to identify defects and inconsistencies.

**Automated Testing:** Automation tools and frameworks can be used to automate repetitive test cases, regression tests, and scenarios. Automated black box testing helps improve efficiency and consistency in testing.

**Test Case Design:** Testers design test cases based on requirements, specifications, user stories, and other external sources of information. Test cases cover various functional and non-functional aspects of the software.

**Test Oracles:** Testers use test oracles to determine whether the observed behavior of the software matches the expected behavior. Oracles can be predefined expected outcomes, specifications, or user expectations.

**Test Coverage Analysis:** Test coverage tools help assess the adequacy of test cases by measuring the extent to which they exercise different parts of the software, such as code branches, paths, and functionalities.

## **Advantages of Black Box Testing:**

**User-Centric:** Black box testing focuses on evaluating the software from the perspective of end-users, ensuring that it meets their needs, expectations, and usability requirements.

**Independence:** Testers do not need access to the source code or knowledge of internal implementation details, allowing for independent testing and verification of the software.

**Requirement Validation:** Black box testing helps validate that the software meets specified requirements, functional specifications, and user stories, ensuring compliance with project goals.

**Effective Error Detection:** By focusing on the software's external behavior, black box testing helps identify defects, errors, and inconsistencies that may not be apparent from a code-level inspection alone.

**Real-World Simulation:** Test scenarios and cases in black box testing simulate real-world usage scenarios, providing valuable insights into how the software performs in actual user environments.

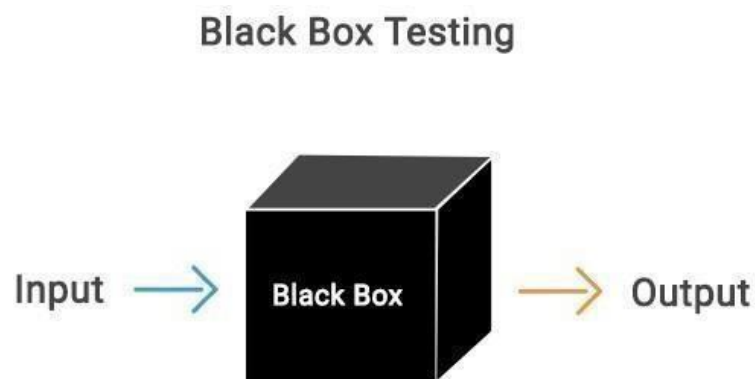


Fig.5.10.4 Black Box testing

#### 5.10 TESTCASES:

S.NO	INPUT	If available	If not available
1.	User signup	signup process completed	There is no process
2.	User Login	user is login	There is no process
3.	View by Voter ID	we got search result from given voter ID	There is no process
4.	Add New Voter	adding new voter details	There is no process
5.	View by Voter ID	search record again	There is no process
6.	Remove Duplicate Data	removing duplicates we got unique records	There is no process
7.	Download Unique Data	download file	There is no process

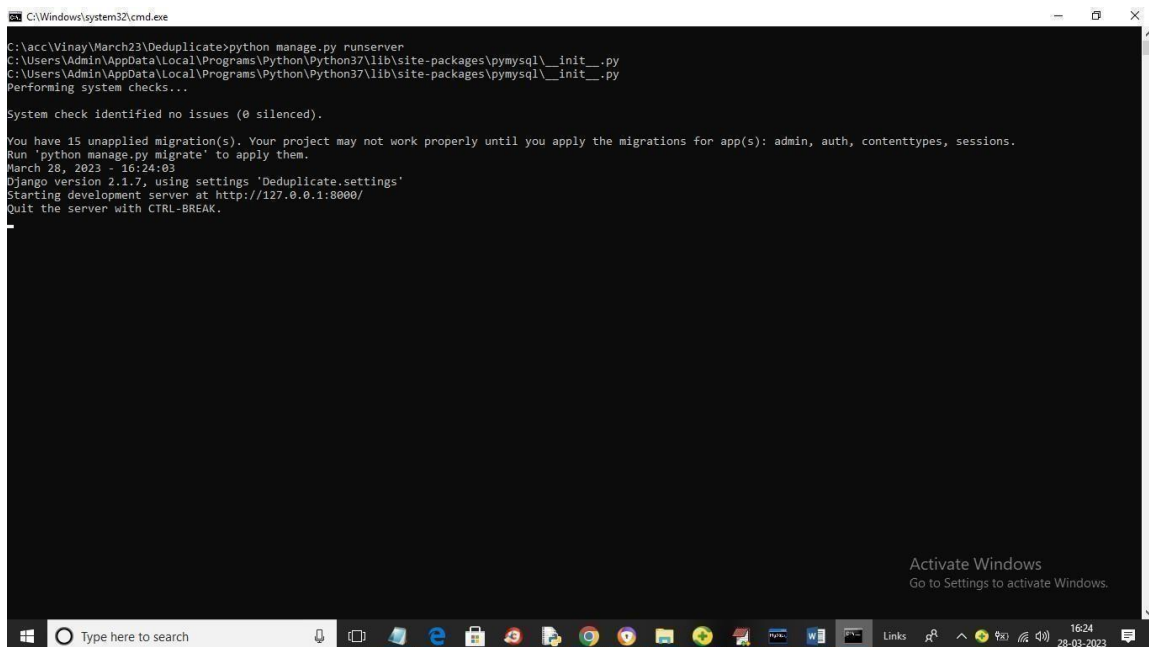
## CHAPTER-6

### RESULTS AND DISCUSSIONS

#### 6.1 Outputs:

To run project install python 3.7.0 and then install all packages given in requirement file and then install MYSQL database and give password as 'root' and after installation open MYSQL and copy content from DB.txt file and paste in MYSQL to create database.

Now double click on 'run.bat' to start python web server and get below page.



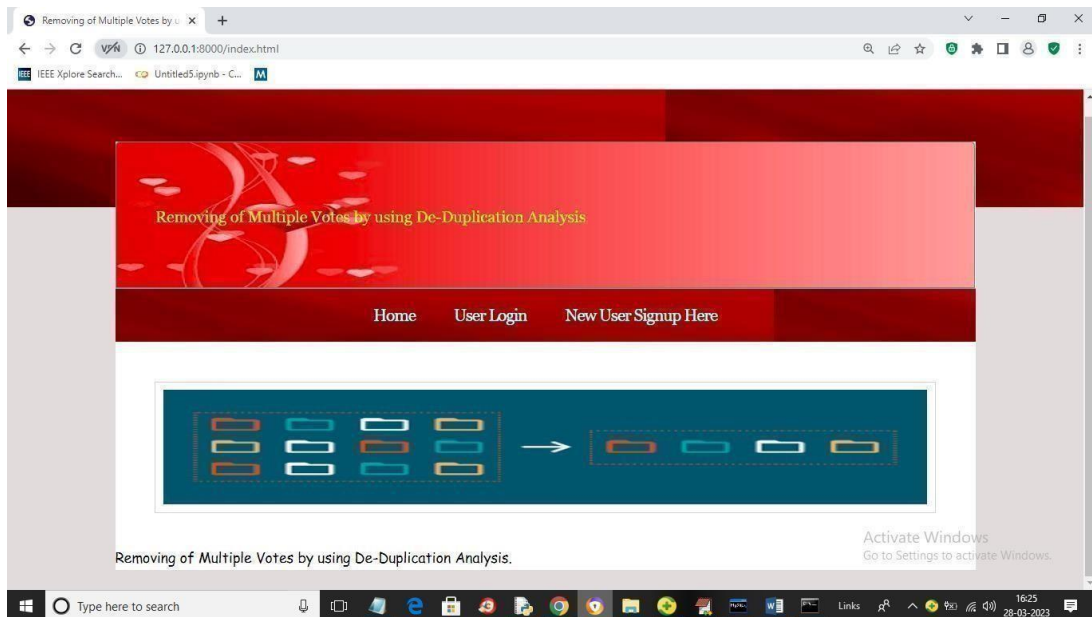
```
C:\Windows\system32\cmd.exe
C:\acc\Vinay\March23\Deduplicate>python manage.py runserver
C:\Users\Admin\AppData\Local\Programs\Python\Python37\lib\site-packages\pymysql\__init__.py
C:\Users\Admin\AppData\Local\Programs\Python\Python37\lib\site-packages\pymysql\__init__.py
Performing system checks...

System check identified no issues (0 silenced).

You have 15 unapplied migration(s). Your project may not work properly until you apply the migrations for app(s): admin, auth, contenttypes, sessions.
Run 'python manage.py migrate' to apply them.
March 28, 2023 - 16:24:03
Django version 2.1.7, using settings 'Deduplicate.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
```

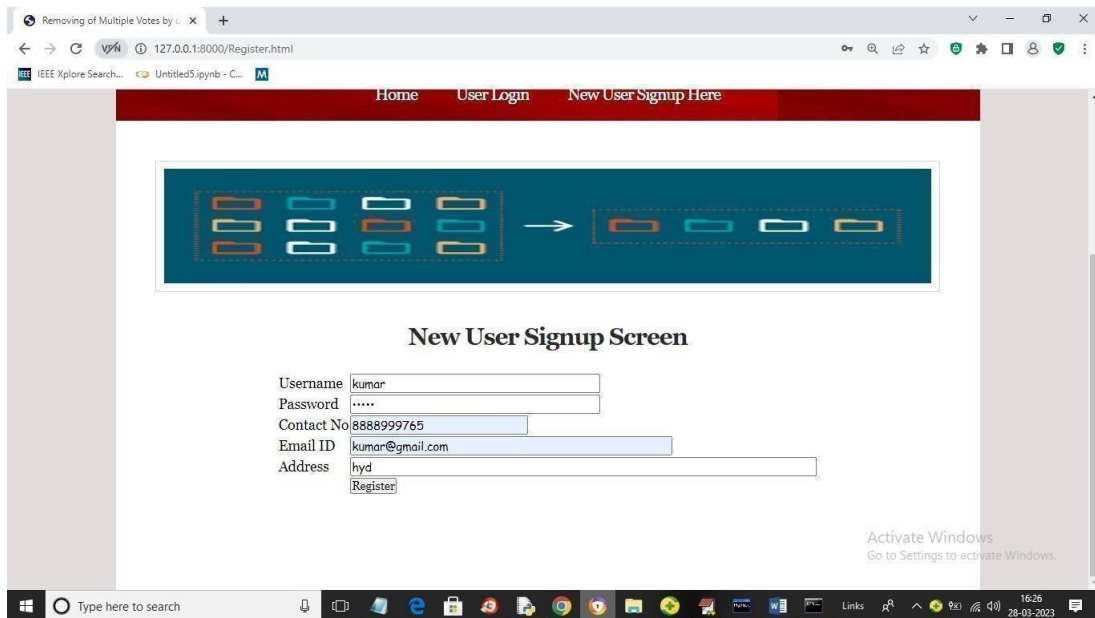
**Fig 6.1.1: Command prompt**

- In above screen server started and now open browser and enter URL as <http://127.0.0.1:8000/index.html> and press enter key to get below page.
- After writing the code in visual studio or pycharm we need to run the code after debugging and running the code it redirects to the command prompt
- we will get a http server url in the command prompt copy the url and paste in web server it redirects to the front end page with is the actual output



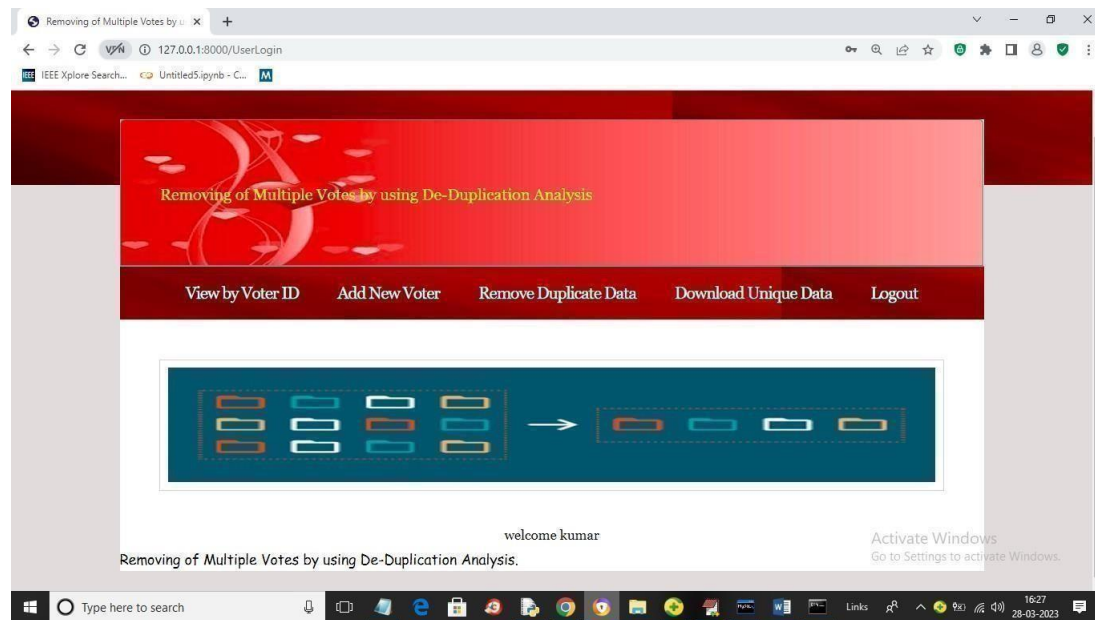
**Fig-6.1.2: Interface**

- After clicking the url which has been appeared in the command prompt
- We will get the above interface which consists of home user login and New user sign up



**Fig-6.1.3: New user signup**

- In the above screen the user can sign in by clicking on sign in option .
- While sign in the new user need to mention few details like username , password Contact no , email id , address.



**Fig-6.1.4: User login**

- The above screen shows us the login of the user.
- User sign have many options like view voterid , add new voter, remove duplicate data Download unique data , logout.
- In above screen user can click on 'View byVoter ID' link to get below page.
- In above screen I entered some 'Voter ID' and then press button to get below page,
- The user will be selected by the admin
- If the admin give access to the user to make any changes then only the can modify the data

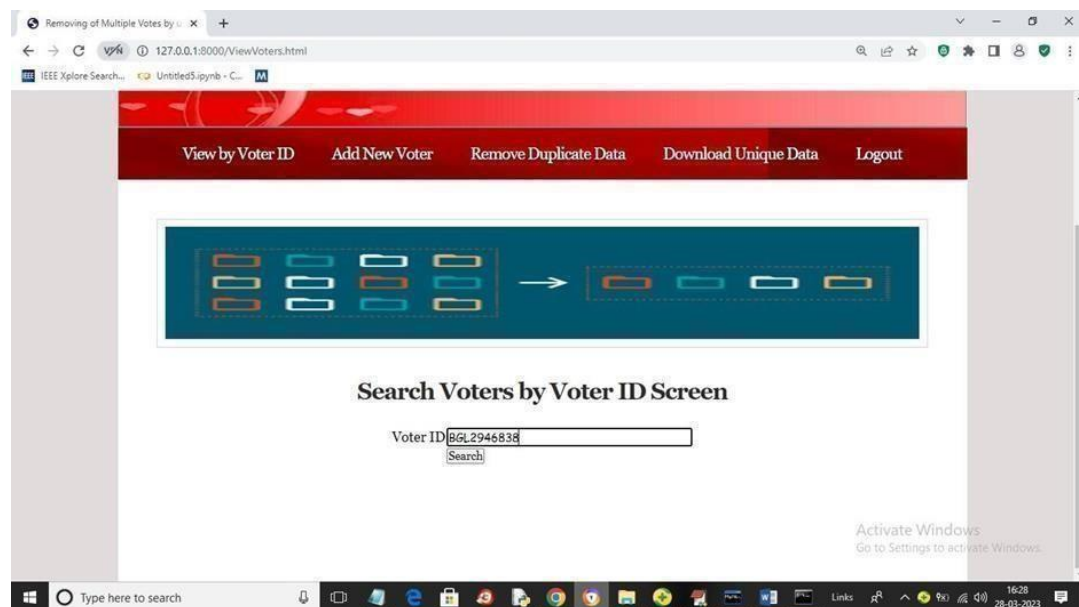


Fig-6.1.5 Search voters

- In the above screen we can see that the user has access to search votes.
- The user can search the voters by name or through the voter id.

Name	Father/Husband	Age	Gender	Voter ID	Aadhar No
Gundreddy Mlikarjuna	Byreddy	66	M	BGL2946838	9444-0032-8237
SreepathyReddyLodugu	RangaReddy	47	M	BGL2946838	8547-7467-3309
RavindranathReddyVenkataDasari	AnkiReddyVenkataDasari	44	M	BGL2946838	9676-8425-9855
VenkataSudhakarReddyAnkiReddy	VenkataReddy	38	M	BGL2946838	7075-5653-9852
VanganurKalavathi	LakshmiNarayanaReddy	50	F	BGL2946838	9441-8178-4479
KoppelaGuruLakshmi	KoppelaPaamuleti	22	F	BGL2946838	9550-6988-6575
Venkatamma Talari	Ramulu Talari	34	F	BGL2946838	8106-8181-6733
Krishna	Venkatanarasaya	59	M	BGL2946838	9100-8634-1950

Fig-6.1.6 voters details

- In the above screen we can see the all the details of the voters.
- The details consists of the name , father name, age , gender, voter id Aadhar no etc .

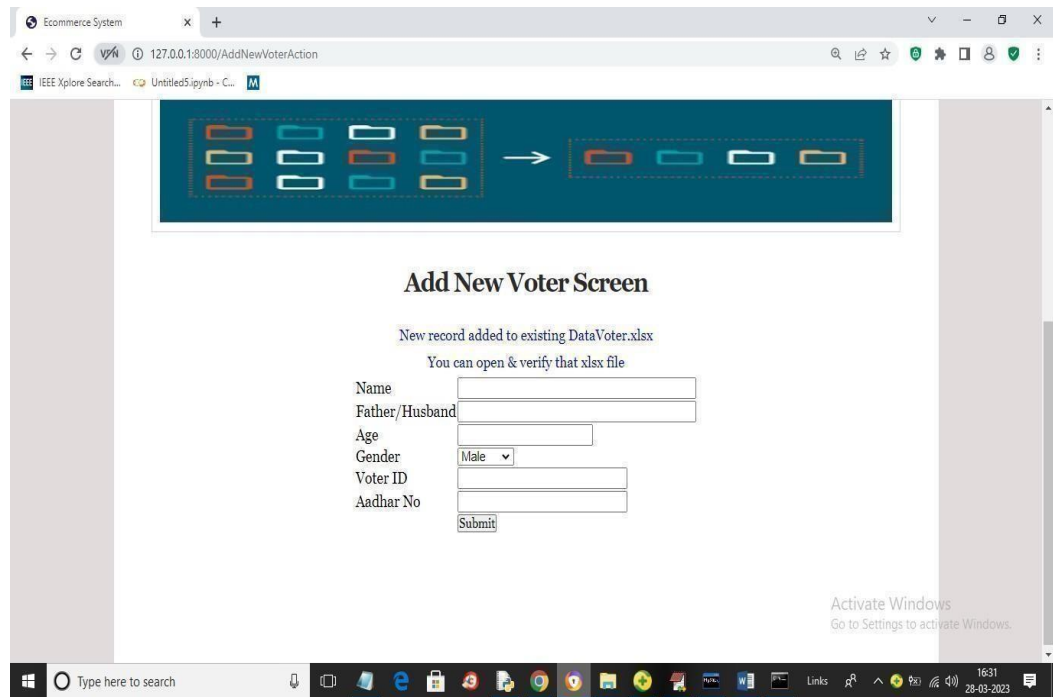
**Add New Voter Screen**

Name	<input type="text" value="Kumar"/>
Father/Husband	<input type="text" value="Rajesh"/>
Age	<input type="text" value="34"/>
Gender	<input type="text" value="Male"/>
Voter ID	<input type="text" value="543"/>
Aadhar No	<input type="text" value="987612347654"/>
	<input type="button" value="Submit"/>

**Fig-6.1.7 Add new voter screen**

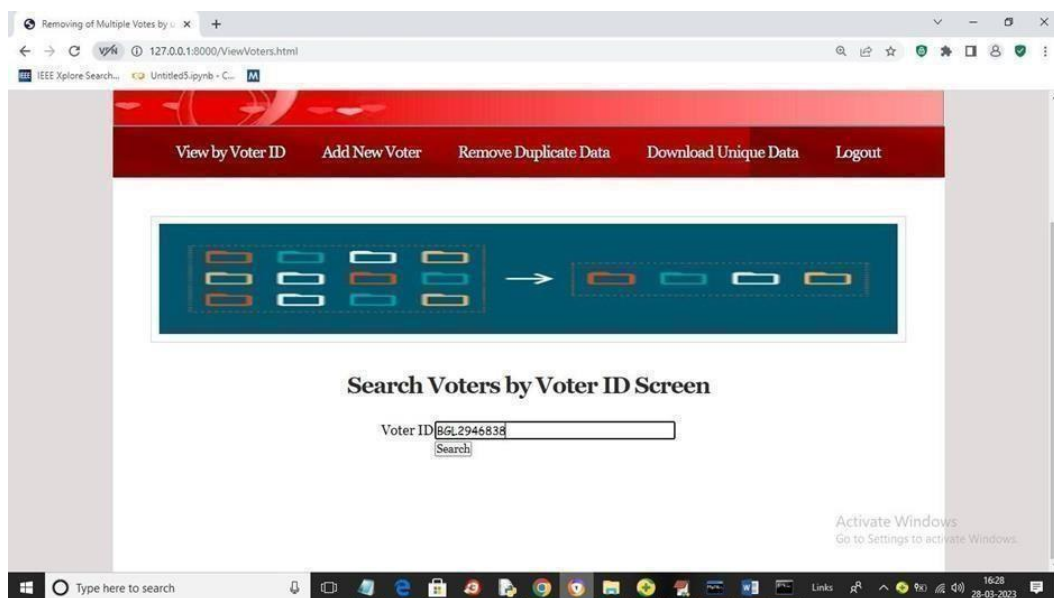
- In above screen adding new voter details and then press button to add voter to excel file and get below output and the given voter id is 543 and by giving this ID we can search record
- While adding the new voter id we need to provide all the details.
- The details includes age, id , aadhar no etc.
- This addition of voters can be done only through admin and the user only.
- Only the admin has access to choose the user.





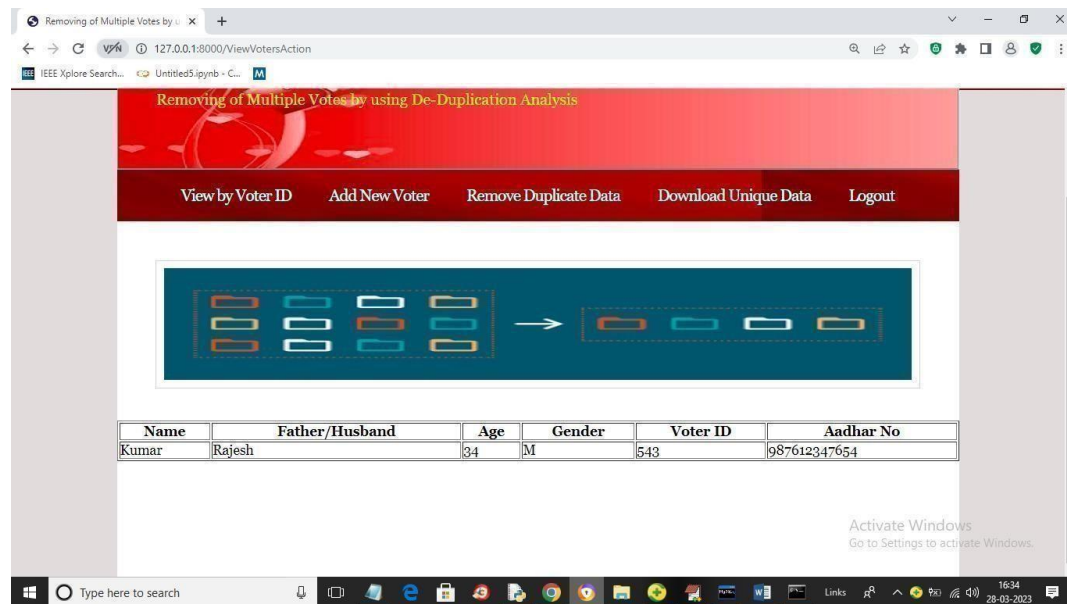
**Fig-6.1.8 Add new voter screen**

- After the user has updated the newly added .
- A new record has been added to the existing data voter list.
- We can open and verify the details of the newly added voter.



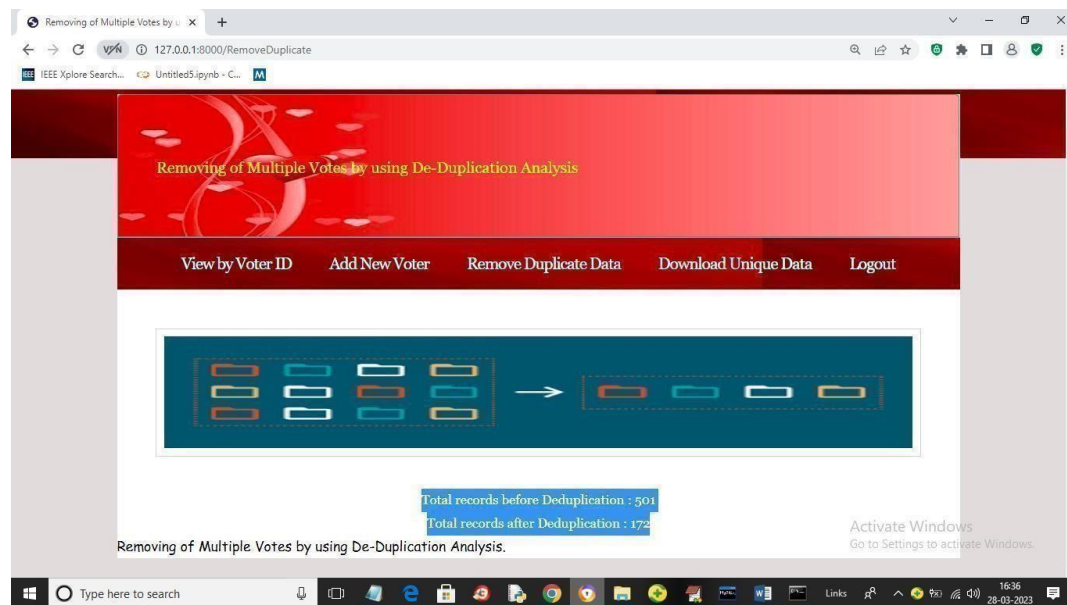
**Fig-6.1.9 Search voters**

- In the above screen we can see the details of the newly added voter by giving the correct voter id.
- After giving the correct voter id we can get the details of the newly added voter data.



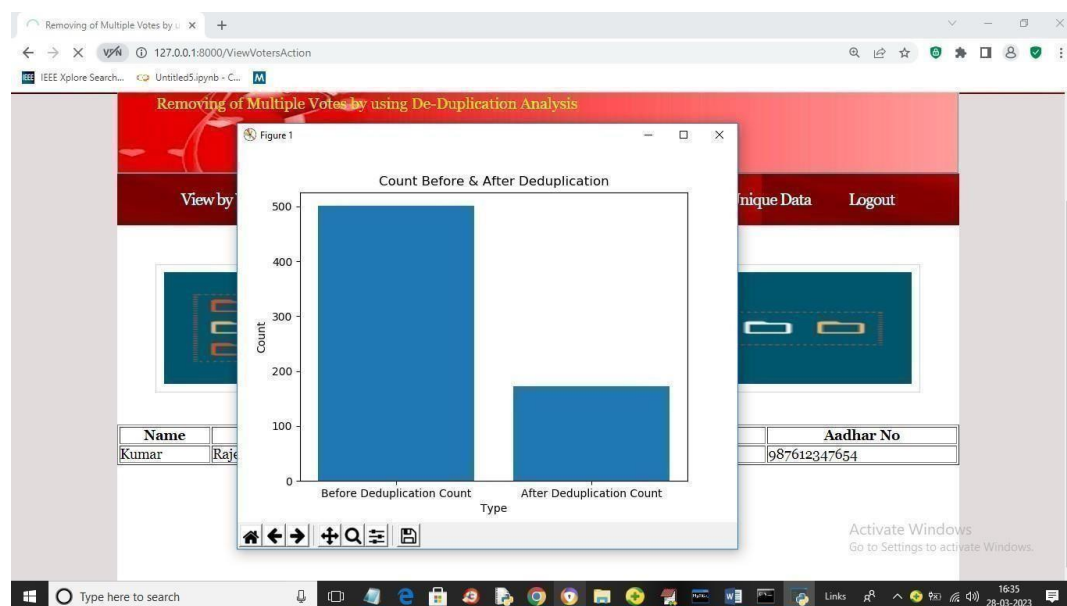
**Fig-6.1.10 Data of added voter**

- In the above screen we can see the details of the added voter.
- If we see the details then the new data is added if not the data is not yet added to the list.
- If we want to remove the data we can remove the clicking on the removeduplicant data.
- This removal can be done only if there is any similar data appears in the list.
- Removing of data is done by the admin and the admin accessed users only.

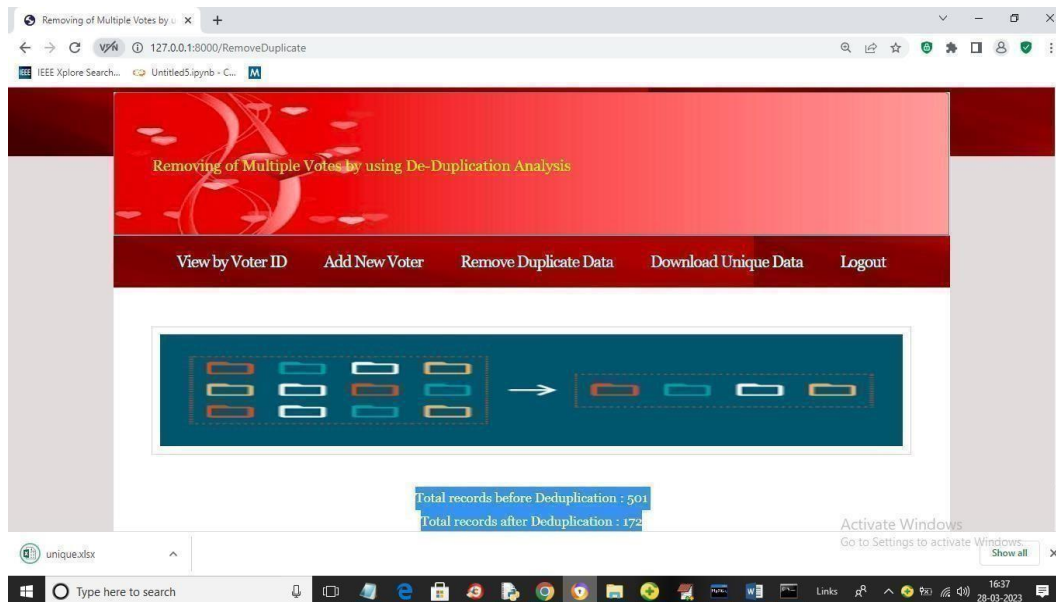


**Fig-6.1.11 Removing voter data**

- We can also delete the voter data this is done to ensure that there is no Duplicate values present in the data.



**Fig-6.1.12 Graph before and after removing duplicate data**



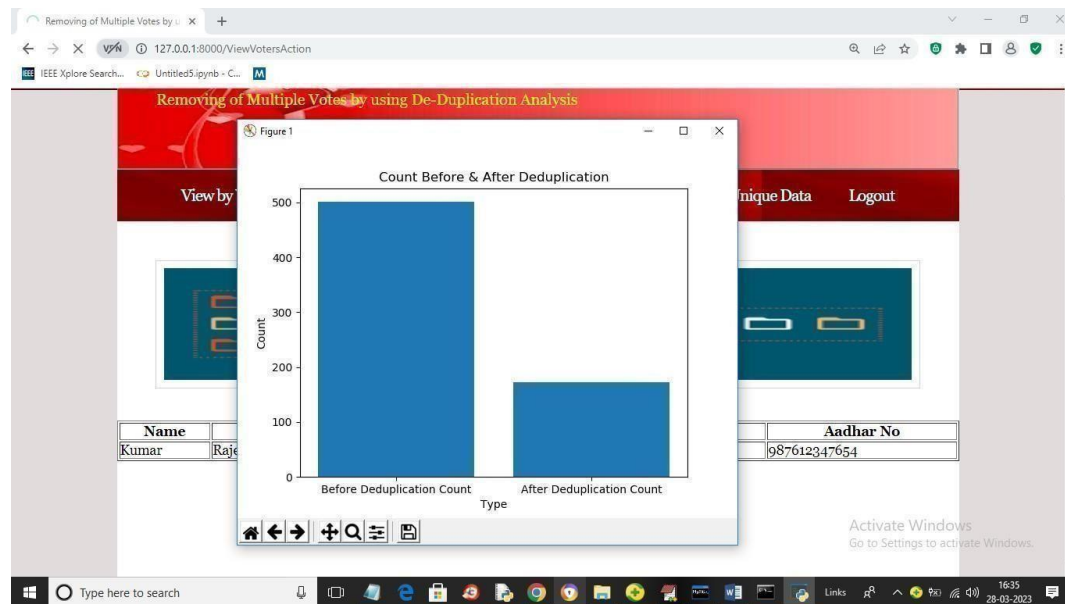
**Fig-6.1.13 Download the data**

- In the above screen we can see the total records before duplication and after duplication.
- Before duplication we have total 501 records and after duplication we have 172 records.
- Clearly we can say that the duplicant values have been deleted or eliminated

	Name	Gender	ID
163	GUNDA S/GUNDA R	23 F	BGL29454;9137-2231-1991
164	MANIBHA RAMALIN	21 F	BGL29489;7219-7012-8813
165	KATAM P/KATAM S	22 M	RGS10208;9112-1722-6002
166	KAMATAN/KAMATAN	26 F	RGS11152;9103-9734-9210
167	KAMMIU I/KAMMIU I	25 F	AP231580;9414-3354-6776
168	KANAKAL/KANAKAL	28 M	BGL29489;9407-9061-3688
169	KANCHAR/KANCHAR	24 M	RGS13814;9734-5077-9000
170	KANDULA/KANDULA	23 M	BGL29470;9908-7183-5121
171	KANISSETT/KANISSETT	23 M	AP231580;9785-0791-7735
172	KARANAN/KARANAN	22 M	BGL29451;9778-1102-2277
173	Kumar Rajesh	34 M	543 987612347654

**Fig-6.1.14 Data after removing duplication**

- After removing the duplicates values the size of the comes to 172
- Which means all the duplicate data has been removed.



**Fig-6.1.15 Graph before and after removing duplication**

- The above graph shows about the data before and after duplication analysis.
- Clearly we can see that there is huge decrease of data after removing Similar values.

## 6.2 Domains and areas where deduplication is critical :

Deduplication, the process of removing duplicate or redundant data from a dataset, is a crucial technique in various domains to improve efficiency, reduce storage costs, and ensure data accuracy. However, there are several areas or domains where deduplication can face challenges or may not be suitable:

**Critical Systems with Unique Identifiers:** In systems where each data entry must have a unique identifier, such as primary keys in databases, blindly applying deduplication can lead to the loss of critical data or system integrity issues if not handled carefully.

**Highly Variable Data:** Deduplication becomes challenging when dealing with data that exhibits significant variability or entropy, making it difficult to define what constitutes a duplicate. This includes data like natural language text, multimedia files, or sensor data with high levels of noise or randomness.

**Encrypted or Compressed Data:** Deduplicating encrypted or compressed data can be problematic because even minor changes in the data can result in vastly different representations, making it challenging to identify duplicates without decrypting or decompressing the data first, which may not always be feasible or desirable.

**Data with Privacy or Security Constraints:** Deduplication may conflict with privacy or security requirements, particularly in sensitive domains like healthcare or finance, where strict regulations govern data handling. Removing duplicates may inadvertently expose confidential information or violate privacy regulations.

**Data with Temporal or Contextual Variability:** Data that evolves over time or within different contexts may pose challenges for deduplication. For example, in social media data, the same content may be duplicated across multiple users' timelines but have different contextual relevance or timestamps.

**Partial Matches and Fuzzy Deduplication:** Identifying duplicates based on partial matches or similarity thresholds (fuzzy deduplication) can be complex, especially when dealing with noisy or incomplete data, such as names, addresses, or product descriptions with variations in spelling, formatting, or abbreviations.

**Distributed or Federated Data Sources:** In distributed systems or federated databases, deduplication across multiple data sources or replicas can be challenging due to network latency, synchronization issues, and the need to maintain consistency and coherence across distributed copies of the data.

**Legal and Compliance Considerations:** Deduplication may be constrained by legal requirements or industry regulations that mandate data retention or prohibit the modification or deletion of certain records, especially in legal proceedings, auditing, or archival contexts.

**Performance Overhead and Scalability:** Depending on the deduplication algorithm and implementation, the process can impose significant computational overhead and resource requirements, making it impractical or inefficient for large-scale or real-time applications without careful optimization and scalability considerations.

**Cultural or Semantic Variability:** Cultural differences, language nuances, and semantic variations in data can complicate deduplication efforts, especially in multinational or multicultural contexts where the same concept may be expressed differently across different languages or dialects.

### 6.3 Domains where deduplication is used most :

Deduplication finds utility across various domains where managing large volumes of data efficiently is essential. Some of the domains where deduplication is commonly used include:

**Data Storage and Backup:** Deduplication is extensively employed in storage systems and backup solutions to reduce storage requirements by eliminating duplicate copies of data. This is particularly beneficial in environments with frequent backups or where storage costs are a concern.

**Data Archiving and Retrieval:** Archiving systems often utilize deduplication to store and manage historical data efficiently. By removing redundant data, deduplication helps optimize storage space and streamline data retrieval processes.

**Cloud Computing and Object Storage:** Deduplication is integral to cloud storage services and object storage systems, enabling providers to offer scalable, cost-effective storage solutions to customers while maintaining high performance and reliability.

**Content Delivery Networks (CDNs):** CDNs leverage deduplication to cache and serve content efficiently to users by identifying and storing only unique or non-redundant content across distributed edge servers. This enhances content delivery speed and reduces bandwidth usage.

**Email and File Servers:** Deduplication is commonly applied in email servers and file storage systems to eliminate duplicate attachments, messages, or files, thereby optimizing storage capacity and improving data management workflows.

**Document Management and Collaboration Platforms:** Document management systems and collaboration platforms employ deduplication to ensure that multiple users working on shared documents or projects do not inadvertently create duplicate copies, thus maintaining data consistency and reducing clutter.



**Database Management Systems:** Deduplication techniques are used in database management systems (DBMS) to optimize storage and improve query performance by eliminating redundant or duplicate records within databases.

**Virtualization and Data Center Infrastructure:** Virtualization platforms and data center infrastructure often employ deduplication to minimize the storage footprint of virtual machine images, snapshots, and backups, thereby maximizing resource utilization and reducing costs.

**Data Integration and ETL Processes:** Deduplication is crucial in data integration and extract, transform, load (ETL) processes to ensure data quality and consistency across disparate sources by identifying and resolving duplicate records or inconsistencies.

**Digital Media and Content Management:** In digital media and content management systems, deduplication helps streamline media asset libraries by identifying and removing duplicate images, videos, or multimedia content, enhancing content organization and accessibility.

## **CHAPTER-7**

### **CONCLUSION**

There is a rapid increase in size of data which lead to heat energy consumption, duplicate data, and inconsistent data organization. The framework in order to fulfil a balance between changing storing efficiency and performance improvement in system. The proposed system is capable of handling scalability problem by removing duplicate data. Deduplication aids in saving the storage space. This project helps in easy maintenance of data so that no duplicate files are saved. It works for text, images, audio and video. With the evolution, storage resources of commodity machines can be efficiently utilized.

The exponential growth of data in recent years has presented significant challenges in terms of storage efficiency, energy consumption, and data organization consistency. This surge in data size has led to issues such as increased heat energy consumption, the proliferation of duplicate data, and inconsistent data organization, posing formidable obstacles to system performance and scalability.

In response to these challenges, this proposes a comprehensive framework aimed at achieving a delicate equilibrium between enhancing storage efficiency and improving system performance. Central to this framework is the implementation of a sophisticated system capable of effectively addressing scalability issues by systematically identifying and eliminating duplicate data. This process of deduplication not only optimizes storage space but also streamlines data maintenance procedures, ensuring that redundant files are systematically identified and removed across various data formats including text, images, audio, and video.

By leveraging the proposed framework, organizations stand to benefit from streamlined data management processes, reduced storage overheads, and improved utilization of storage resources, even within commodity hardware environments. As data storage requirements continue to evolve, the ability to efficiently manage and utilize storage resources becomes increasingly critical. Therefore, the proposed framework represents a significant step forward in addressing the challenges posed by the exponential growth of data while simultaneously facilitating the efficient utilization of storage resources to meet the evolving needs of modern data-driven enterprises.

## **7.1 Future Scope :**

The future scope of deduplication analysis, particularly in the context of removing multiple data, is quite promising and can have several potential developments and applications:

With the ever-increasing volume of data generated by organizations, there's a growing need for efficient data management techniques. Deduplication analysis can play a significant role in data warehousing and big data environments by reducing storage costs and improving data processing efficiency.

As data privacy regulations become more stringent worldwide (such as GDPR, CCPA), organizations need to ensure compliance while managing large datasets. Deduplication analysis can help organizations identify and eliminate duplicate or redundant data, reducing the risk of data breaches and ensuring compliance with regulatory requirements.

Deduplication analysis can be integrated with machine learning and artificial intelligence algorithms to enhance data preprocessing and feature engineering tasks. By removing redundant data, models can be trained more efficiently, leading to better performance and accuracy in predictive analytics and machine learning applications.

Deduplication analysis can facilitate data integration and master data management efforts by identifying and resolving duplicate records across disparate data sources. This can improve data quality, consistency, and reliability, enabling organizations to make better-informed decisions based on clean and accurate data.

Deduplication analysis is a crucial step in data cleaning and preprocessing pipelines, where the goal is to prepare data for analysis or modeling. Automated deduplication techniques, combined with advanced algorithms for record linkage and entity resolution, can streamline the data cleaning process and improve the quality of analytical insights.

In decentralized environments like blockchain and distributed ledgers, deduplication analysis can help maintain data integrity and consensus among network participants. By

identifying and eliminating duplicate transactions or records, deduplication techniques can contribute to the reliability and efficiency of distributed systems.

In CRM systems, deduplication analysis is essential for ensuring a single, unified view of customers across different channels and touchpoints. By removing duplicate customer records, organizations can improve customer data accuracy, enhance marketing targeting efforts, and deliver better customer experiences.

As the demand for real-time analytics and processing increases, deduplication techniques will need to adapt to handle streaming data and high-velocity data streams effectively. Real-time deduplication analysis can help organizations maintain data quality and consistency in dynamic and rapidly changing environments.

With the proliferation of Internet of Things (IoT) devices and edge computing infrastructure, deduplication analysis can be deployed at the edge to reduce data transmission bandwidth and storage requirements. This can improve the efficiency of IoT data processing and enable edge devices to operate more autonomously with limited resources.

Deduplication analysis techniques developed in one domain can be applied and adapted to various other domains, including healthcare, finance, retail, manufacturing, and more. The future scope of deduplication analysis extends beyond specific industries and can have broad applications across diverse sectors, contributing to data-driven decision-making and innovation.

## REFERENCES

- [1] A. Gupta, N. Sharma (2019). Secure and Efficient Deduplication with Privacy Preservation in Cloud Storage
- [2] Baravkar, S., & Mali, V. (2016). Authorized data Deduplication using Hybrid Cloud Technique. Year of Publication: 2016
- [3] Chen, J., & Wu, X. (2018). Blockchain-Based Data Deduplication Scheme for Cloud Storage Security. IEEE Transactions on Services Computing, 11(4), 714-724.
- [4] Chen, Y., & Li, Z. (2019). Fuzzy Logic-Based Data Deduplication Method for Cloud Storage Systems. International Journal of Fuzzy Systems, 21(6), 1912-1925.
- [5] Ezeife, C. I. (2007). The use of smart tokens in cleaning integrated warehouse data. Year of Publication: 2007.
- [6] Festus, O. A. (2016). Data finding, sharing and duplication removal in the cloud using file checksum algorithm. Year of Publication: 2016.
- [7] Gupta, R., & Singh, S. (2019). Dynamic Data Deduplication Technique for Cloud Storage Systems. International Journal of Computer Applications, 178(4), 35-41.
- [9] Gupta, S., & Jain, A. (2017). Performance Analysis of Data Deduplication Techniques in Cloud Storage Environments. International Journal of Computer Applications, 169(3), 33-40
- [10] Huang, T., & Zhang, Q. (2019). Scalable and Efficient Data Deduplication Technique for Cloud-Based Backup Services. Journal of Supercomputing, 75(8), 5076-5091.
- [11] J. Smith, K. Johnson (2016). Efficient Data Deduplication in Cloud Storage Systems.

- [12] Kumar, A., & Sharma, S. (2018). Hybrid Approach for Data Deduplication in Cloud Storage. *International Journal of Computer Applications*, 182(2), 9-15.
- [13] Kumar, S., & Gupta, R. (2019). Robust Data Deduplication Scheme for CloudStorage with Hybrid Encryption. *Information Sciences*, 502, 354-369.
- [14] Kassirer & Angell (2010), multiple reports of the same observations can over emphasize the importance of the findings, overburden busy reviewers
- [15] Li, H., & Wang, Q. (2019). Privacy-Preserving Data Deduplication in Cloud Storage Using Homomorphic Encryption. *IEEE Transactions on Information Forensics and Security*, 14(12), 3145-3157.
- [16] Li, W., & Wang, Y. (2018). Efficient Data Deduplication Techniques for Large-Scale Cloud Storage Systems. *Future Generation Computer Systems*, 87, 643-655.
- [17] Liu, Z., & Chen, X. (2020). Performance Evaluation of Data Deduplication Techniques in Cloud Storage Environments. *International Journal of Grid and Utility Computing*, 11(5), 586-598.
- [18] Maragatharajan, M., & Prequiet, L. (2017). Removal of Duplicate data from Encrypted Cloud Storage. Year of Publication: 2017.
- [19] Osuolale, Festus (2016). Data finding, sharing and duplication removal in the cloud using file checksum algorithm
- [20] Pasquale Puzio; Refik Molva; Melek Önen; Sergio Loureiro (2013) . Clouded up: Secure Deduplication with Encrypted data for Cloud Storage
- [21] Puzio, P., Molva, R., Önen, M., & Loureiro, S. (2013). Clouded up: Secure Deduplication with Encrypted data for Cloud Storage. Year of Publication: 2013.
- [22] Radhakrishnan, R., & Kumar, S. (2018). Efficient Data Deduplication in Cloud Storage Systems. Year of Publication: 2018.

- [23] Snehal Baravkar , Vaishali Mali (2016) . Authorized data Deduplication using Hybrid Cloud Technique.
- [24] Wang, L., & Chen, J. (2019). Cost-Effective Data Deduplication Techniques for Cloud Storage Systems. IEEE Access, 7, 55684-55695.
- [25] Zhang, H., & Liu, L. (2019). Distributed Data Deduplication Techniques for Cloud Based Storage Services. Journal of Parallel and Distributed Computing, 123, 84-95.
- [26] Zhang, X., Zhang, Y., & Zhang, Z. (2018). Efficient and Secure Data Deduplication Scheme in Cloud Storage Systems. IEEE Transactions on Cloud Computing, 6(3), 689- 701.
- [27] Zhang, Y., & Wu, W(2018) . Data Deduplication Techniques in Cloud Storage: A Comprehensive Survey