USE CASE STUDY REPORT

Guardians of Ethics – Anti-Plagiarism Tracking Management System

Group No: Group 19

Student Names: Hashwanth Moorthy & Moheesh Kavitha Arumugam

Executive Summary:

This concept arises as a timely response to the dual nature of modern-day high-speed internet connection. While it promotes knowledge development, it also presents obstacles, notably in retaining the creative integrity of educational initiatives. This initiative intends to address the widespread problem of plagiarism, particularly among students in academic institutions. The objective is to create a sophisticated system that records and controls plagiarism incidents, ensuring that students are properly acknowledged for their genuine contributions while preserving intellectual honesty values.

The working theory of the system involves utilizing plagiarism-checking software for each submitted project. Professors evaluate authenticity, and the software generates reports detailing the percentage of plagiarism by referencing external sources. Consequences are tiered based on the plagiarism percentage based on the categories. The relational model is built using the entities that are contributing to plagiarism management. We utilized UML alongside EER diagrams for a more comprehensive modeling approach, as UML covers system architecture and behavior beyond database design, ensuring a unified representation throughout the software development lifecycle. Then a relational schema is created based on the above EER and intertwined with the other entities using the foreign keys.

With the following relational schema in hand, a new database is created keeping in mind the above constraints. The data is collected from the respective university database along with the information of the students and their respective projects. All the details of the funding company and their plagiarism software are also collected and stored in our database for our efficient use of the use case. In the Python implementation, the MySQL connector is utilized to connect to the MySQL server, execute queries, and retrieve results. The implementation includes diverse queries such as analyzing similarity percentages of student reports, counting students under each professor, examining student project participation, investigating plagiarism reports, and evaluating average similarity percentages of students. The results are visualized through Seaborn and Matplotlib, generating informative charts such as histograms, bar plots, pie charts, and line plots. Additionally, the system categorizes students based on their average plagiarism levels, providing insights into the prevalence of plagiarism and facilitating proactive measures to maintain academic integrity. I opted for NoSQL, specifically MongoDB, to handle unstructured data and provide flexibility for evolving data structures, enhancing the project's adaptability and performance. The document-oriented nature of MongoDB aligns with the complex relationships in the application, offering seamless integration with the existing SQL database.

As the project progresses, it envisions not only enhancing academic honesty but also providing a scalable and adaptable solution for educational institutions across diverse disciplines. The next steps involve pilot implementations, gathering feedback for refinements, exploring advanced technologies for improved detection, and collaborating with educational authorities to position the system as a comprehensive tool for promoting and preserving academic integrity in the digital age. In summary, this initiative stands as a beacon for the promotion of originality and ethical conduct within educational settings, ensuring that the pursuit of knowledge remains a noble and authentic endeavor.

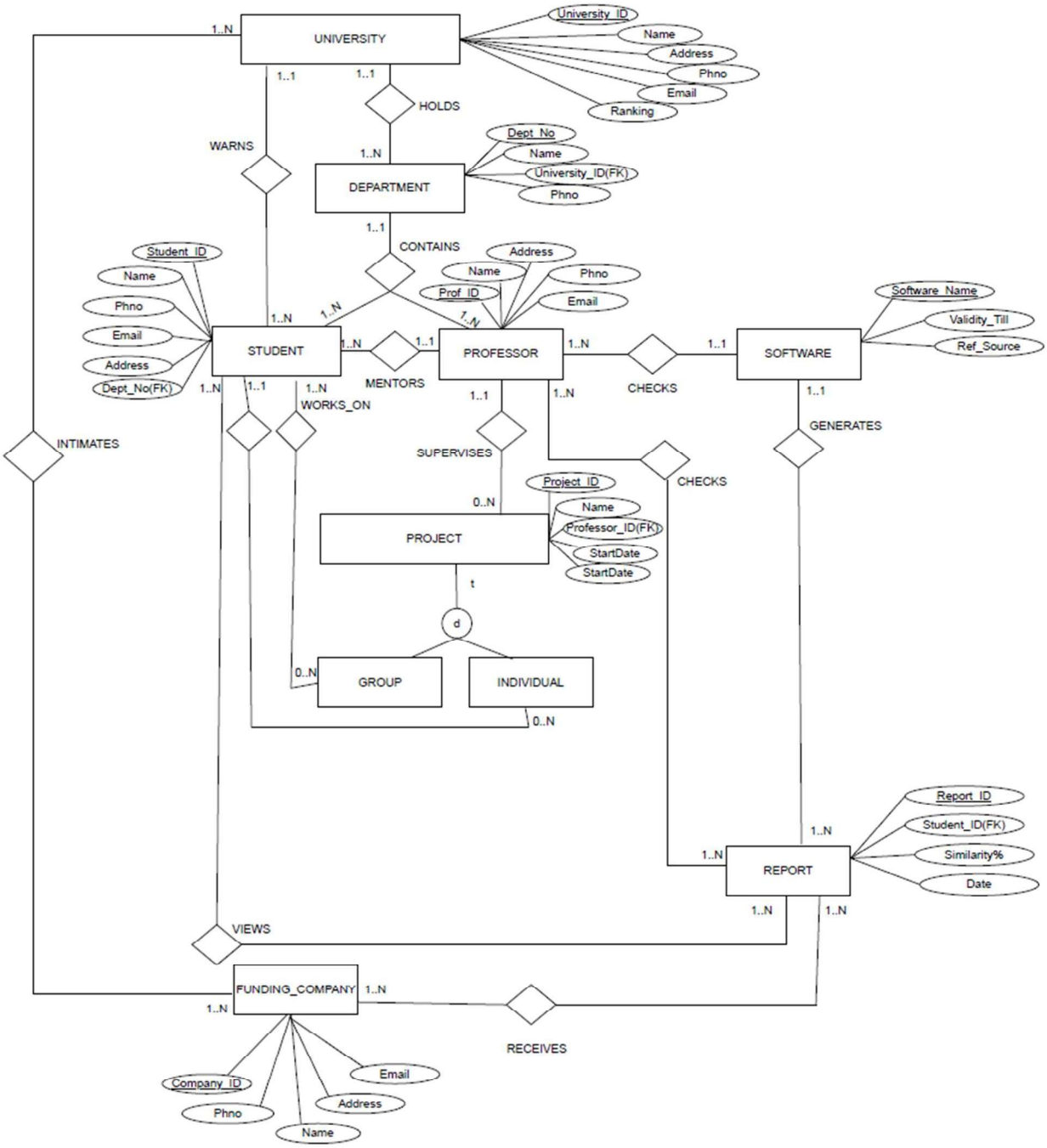
I. Introduction

In the current digital age, where high-speed internet access has made data readily available, it brings both advantages and drawbacks. While it offers opportunities for expanding knowledge, it simultaneously poses challenges, particularly hindering the creative process. This contrast is particularly relevant for students in educational institutions, emphasizing the necessity for their work to be distinct and original. Fostering a culture of intellectual integrity becomes paramount, creating a positive environment that prioritizes honesty. Plagiarism emerges as a serious concern, prompting the need for a comprehensive system to track the plagiarism history of every student within a university. Our approach not only evaluates the extent of plagiarism over time but also emphasizes tailored consequences based on the percentage of plagiarism committed, ensuring a fair and educational response to promote academic integrity. Additionally, implementing such a system not only safeguards the value of students' sincere efforts but also contributes to a more ethical and transparent educational environment. This proactive measure encourages a deeper understanding of academic responsibility and provides educators with valuable insights for implementing targeted interventions to prevent plagiarism. It also reinforces a commitment to fostering a learning environment that values originality and supports the growth of students as ethical contributors to their fields of study.

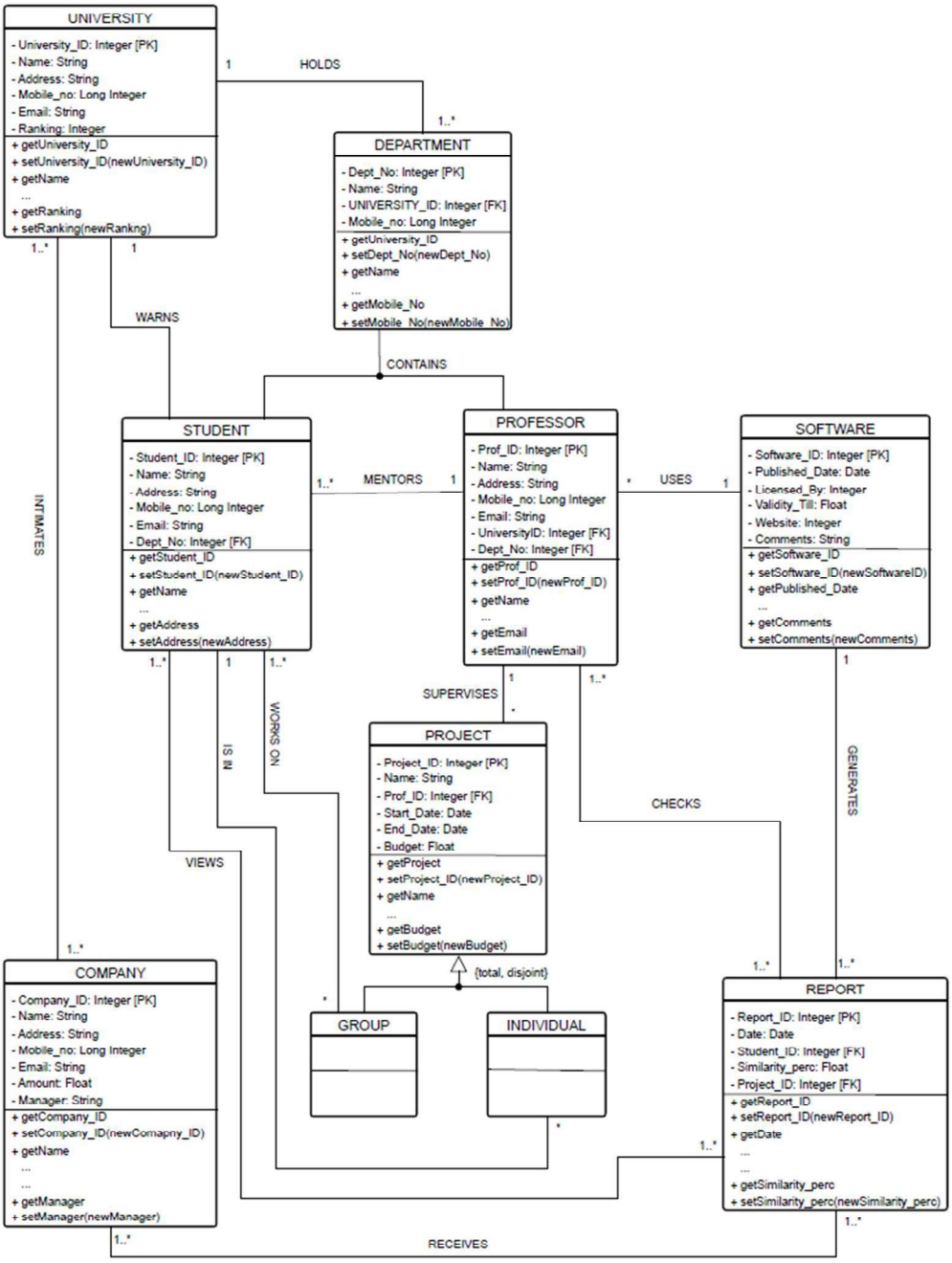
Implementing the plagiarism tracking project within a relational database offers numerous advantages. The structured tables and entities efficiently organize diverse data concerning students, professors, projects, and plagiarism reports. This ensures data integrity through referential constraints, making it easier to retrieve precise information. The scalable architecture accommodates the dynamic nature of academic institutions, allowing for seamless growth as the institution expands. Access controls enhance data security, protecting sensitive information from unauthorized access. Historical data tracking, helps in-depth analysis of changes over time. Aggregated reporting simplifies the interpretation of academic integrity trends, providing valuable insights for decision-making. With a centralized repository, redundancy is minimized, and consistency is maintained across all records. The modular design of the relational database facilitates ease of maintenance and updates, ensuring adaptability to changing academic requirements without causing disruptions. The relational database approach presents a comprehensive and efficient solution tailored to the specific needs of tracking plagiarism in an academic environment. Its user-friendly features and flexibility make it an optimal choice for managing academic integrity and ensuring the accuracy and security of data related to students, professors, projects, and plagiarism reports

II. Conceptual Data Modeling

1. EER Diagram



2. UML Diagram



III. Mapping Conceptual Model to Relational Model

Primary Key- Underlined **Foreign Key- *Italicized***

- University (UniversityID, Name, Address, PhnNo, Email, Ranking)

- Department (DeptNo, Name, PhnNo, *UniversityID*)
  - *UniversityID* is a foreign key of University relation and NOT NULL

- Student (StudentID, Name, PhnNo, Email, Address, *UniversityID*, *DptID*, *MentorID*)
  - *UniversityID* is a foreign key of University relation and NOT NULL
  - *DeptID* is a foreign key of Department relation and NOT NULL
  - *MentorID* is a foreign key of Professor relation and NOT NULL

- Professor (ProfID, Name, Address, PhnNo, Email, *DptID*, *SoftwareName*)
  - *DeptID* is a foreign key of Department relation and NOT NULL
  - *SoftwareName* is a foreign key of Software relation and NOT NULL

- Software (SoftwareName, Validity_Till, Ref_Source)

- Report (ReportID, Similarity%, Date, *SoftwareName*)
  - *SoftwareName* is a foreign key of Software relation and NOT NULL

- FundingCompany(CompanyID, Name, Phone, Address, Email)

- Project (ProjectID, Name, StartDate, EndDate, *ProfID*)
  - *ProfID* is a foreign key of Professor relation and NOT NULL

- GroupProject (*GroupProjID*)
  - *GroupProjID* is a foreign key of Project relation and NOT NULL

- Individual (*IndProjID, StudentID*)
  - *IndProjID* is a foreign key of Project relation and NOT NULL
  - *StudentID* is a foreign key of Student relation and NOT NULL

- Intimates (*UniversityID*, *CompanyID*)
  - *UniversityID* is a foreign key of University relation and NOT NULL
  - *CompanyID* is a foreign key of Company relation and NOT NULL

- DepartmentContains (*StudentID*, *ProfID*)
  - *StudentID* is a foreign key of Student relation and NOT NULL
  - *ProfID* is a foreign key of Professor relation and NOT NULL

- StudentWorksOn (*GroupProjID*, *StudentID*)

  - *GroupProjID* is a foreign key of Project relation and NOT NULL
  - *StudentID* is a foreign key of Student relation and NOT NULL

- Views (*StudentID*, *ReportID*)
  - *StudentID* is a foreign key of Student relation and NOT NULL
  - *ReportID* is a foreign key of Report relation and NOT NULL

- Checks (*ProfID*, *ReportID*)
  - *ProfID* is a foreign key of Professor relation and NOT NULL
  - *ReportID* is a foreign key of Report relation and NOT NULL

- Receives (*CompanyID*, *ReportID*)
  - *CompanyID* is a foreign key of Company relation and NOT NULL
  - *ReportID* is a foreign key of Report relation and NOT NULL

Semantics Lost

- A department without any students or instructors is possible, according to the relational model.
- Projects may be assigned as either group or individual assignments. Furthermore, certain projects may not be both.
- It is possible for some reports to be unchecked by professors, and for some professors to review no reports.
- Instructors may have failed to mentor at least one student.

IV. Implementation of Relation Model via MySQL and NoSQL

MySQL Implementation:

The database was created in MySQL and the following queries were performed:

Query 1: Get the list of student who belongs to DeptID='1'(computer science department)

SELECT * FROM student where deptid='1';

StudentID	Name	PhnNo	Email	Address	UniversityID	DeptID	MentorID
13	Ismael Holt	763-728-1358	otgf108@gmail.com	232 East White Fabien Street	243	1	1000039
27	Kathleen Novak	823-091-3848	vwveu6@gmail.com	74 North Green Old Freeway	243	1	1000068
45	Emma Anthony	778-471-1859	pboh@gmail.com	66 East Rocky Old Freeway	243	1	1000068
69	Marc Pope	227-815-6213	owps.nflc@gmail.com	76 South White Milton St.	243	1	1000053
85	Latanya King	546-744-1271	xvtj@gmail.com	34 South Green Milton Drive	306	1	1000018
98	Cassandra Fischer	124-893-4464	adrp267@gmail.com	766 South White Second Road	222	1	1000002
104	Darren Gates	285-588-6253	lyqb11@gmail.com	485 South Green Cowley Freeway	320	1	1000050

Query 2: Get the list of reports which has highest similarity scores

select ReportID, SimilarityPercentage, Date from report where
SimilarityPercentage = (select max(SimilarityPercentage) from report);

ReportID	SimilarityPercentage	Date
69325	28.10	2023-06-18
69572	28.10	2023-02-09
70105	28.10	2023-05-19
70287	28.10	2023-02-09
70352	28.10	2023-12-20
70625	28.10	2023-10-08

Query 3: Get information of students who works on multiple Group Projects

SELECT s.StudentID, s.Name, s.PhnNo, s.UniversityID, s.DeptID, count(*) as 'Count of Projects'
FROM plagiarism.student s
INNER JOIN studentworkson s1 ON (s1.StudentID = s.StudentID)
GROUP BY s.StudentID, s.Name, s.PhnNo, s.UniversityID, s.DeptID, s1.StudentID
HAVING count(*)>1 order by count(*) desc;

StudentID	Name	PhnNo	UniversityID	DeptID	Count of Projects
13	Ismael Holt	763-728-1358	243	1	6
104	Darren Gates	285-588-6253	320	1	6
189	Stephan Arellano	367-538-6876	299	23	6
64	Tamiko Novak	661-211-1157	222	62	5
66	Kellie Galvan	184-165-1624	229	5	5
81	Wallace Savage	305-471-9951	306	5	5

Query 4: Find the university whose students have the highest average similarity score compared to other universities

```
SELECT U.UniversityID, U.Name AS UniversityName, U.Address, U.PhnNo, U.Email, U.Ranking
FROM University U
WHERE U.UniversityID = (
    SELECT UniversityID
    FROM (
        SELECT S.UniversityID, AVG(R.SimilarityPercentage) AS AvgSimilarity
        FROM Student S
        JOIN Views V ON S.StudentID = V.StudentID
        JOIN Report R ON V.ReportID = R.ReportID
        GROUP BY S.UniversityID
        ORDER BY AvgSimilarity DESC
        LIMIT 1
    ) AS UniversityAvgSimilarity
);
```

UniversityID	UniversityName	Address	PhnNo	Email	Ranking
292	Holt School	652 South Green Hague Street	637-744-8238	yjod2@university.org	88

Query 5: Give details of the professor and their department who does not check any report

```
SELECT p.ProfID, p.Name as 'Professor Name', p.DptID, d.Name as 'Department Name'
FROM professor p
LEFT OUTER JOIN checks c ON ( c.ProfID = p.ProfID)
INNER JOIN department d ON ( d.DeptNo = p.DptID)
WHERE c.ReportID is null;
```

ProfID	Professor Name	DptID	Department Name
1000026	Franklin Jones	15	Civil and Environmental Engineering
1000062	Deana Kline	1	Biomedical Engineering
1000063	William Stout	15	Civil and Environmental Engineering
1000095	Wesley Nichols	2	Computer Science and Engineering

Query 6: Retrieve students who have a mentor from the same department and have worked on projects supervised by that mentor

```
SELECT StudentID, Name AS StudentName, Email, S.DeptID, MentorID
FROM Student S
WHERE DeptID IN (
    SELECT P.DptID
    FROM Professor P
    WHERE P.ProfID = S.MentorID
);
```

StudentID	StudentName	Email	DeptID	MentorID
326	Lillian Hoover	uqje1@gmail.com	1	1000062
66	Kellie Galvan	eixy5@gmail.com	5	1000002
56	Paige Cooper	blgu33@gmail.com	15	1000069
156	Rickey Bowman	ihwt552@gmail.com	17	1000008
35	Arnold French	abgwn7@gmail.com	35	1000061
86	Rudy Mc Knight	fmvx.tqsg@gmail.com	36	1000033
72	Jayson Shields	pqlf@gmail.com	36	1000091

Query 7: Find professors who are not mentoring any student

```
SELECT * FROM Professor
WHERE NOT EXISTS (SELECT 1 FROM Student WHERE Student.MentorID = Professor.ProfID);
```

ProfID	Name	Address	PhnNo	Email	DptID	SoftwareName
1000006	Robbie Baird	371 South Green Fabien Way	221-391-6541	unlu.emonl@gmail.com	19	Plagspotter
1000044	Ruth Simmons	62 North Rocky Clarendon Road	653-474-6213	gbti@gmail.com	8	Dustball
1000051	Clifford Diaz	341 North White Second Boulevard	217-031-0412	raoiqt417@gmail.com	33	Plagiarism Checker X
1000079	Ron George	692 West White Clarendon Blvd.	202-430-7092	hgygi.owpxi@gmail.com	57	PlagiarismDetector.net

Query 8: Retrieve a combined list of students who have worked on group projects and individual projects

```
SELECT SWO.StudentID, S.Name AS StudentName, S.Email, S.Address, S.UniversityID, S.DeptID,
S.MentorID
FROM StudentWorksOn SWO
JOIN Student S ON SWO.StudentID = S.StudentID
UNION
SELECT I.StudentID, S.Name AS StudentName, S.Email, S.Address, S.UniversityID, S.DeptID,
S.MentorID
FROM Individual I
JOIN Student S ON I.StudentID = S.StudentID;
```

StudentID	StudentName	Email	Address	UniversityID	DeptID	MentorID
0	Abel Warren	oupo8@gmail.com	87 South White Milton Way	299	1	1000021
2	Janice Payne	lgcf243@gmail.com	83 North Rocky Hague Parkway	208	8	1000025
3	Gretchen Mason	dsxgf0@gmail.com	43 North Rocky Second Drive	285	35	1000001
4	Lawanda Noble	yeip.kdib@gmail.com	84 South Rocky First Freeway	264	13	1000013
5	Robbie Baird	vlpj761@gmail.com	848 South Rocky Cowley Way	327	13	1000016
6	Carla Compton	thfjh41@gmail.com	610 North Green Clarendon Freeway	264	3	1000028
8	Kendra Stevenson	axtbw.lkcc@gmail.com	620 North Green New Blvd.	257	62	1000037

Query 9: Retrieve information about professors and the projects they supervise, including the total number of students on each project

```
SELECT P.ProfID, P.Name AS ProfessorName, P.DptID, GP.ProjectID,
   (
      SELECT COUNT(DISTINCT SWO.StudentID)
      FROM StudentWorksOn SWO
      WHERE SWO.GroupProjID = GP.ProjectID
   ) AS TotalStudentsOnProject
FROM Professor P
```

JOIN Project GP ON P.ProfID = GP.ProfID;

ProfID	ProfessorName	DptID	ProjectID	TotalStudentsOnProject
1000001	Abel Warren	1	7023	3
1000001	Abel Warren	1	7098	5
1000001	Abel Warren	1	7246	1
1000001	Abel Warren	1	7253	5
1000001	Abel Warren	1	7321	2
1000002	Erick Valentine	5	7228	1
1000002	Erick Valentine	5	7236	2

NoSQL Implementation in MongoDb

The following collections were created under the database named ‘plagiarism’

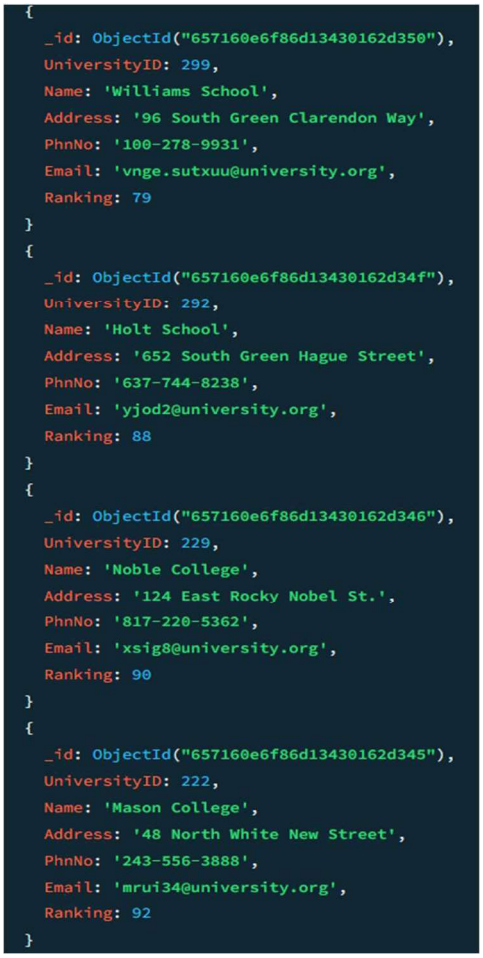
Query 1: Find the maximum similarity Percentage of the report and display its instance

db.report.find().sort({ "SimilarityPercentage": -1 }).limit(1)

```
> db.report.find().sort({ "SimilarityPercentage": -1 }).limit(1)
< {
    _id: ObjectId("657160bcf86d13430162cf4a"),
    ReportID: 69325,
    SimilarityPercentage: 28.1,
    Date: 2023-06-18T00:00:00.000Z,
    SoftwareName: 'PlagScan'
  }
```

Query 2: Find the top 10 universities based on the Ranking

db.university.find().sort({ Ranking: 1 }).limit(10)



Query 3: Get the software name which is expired after January 2015

```
db.software.find({
    Validity_Till: { $gte: ISODate("2015-01-01") }
}, {
```

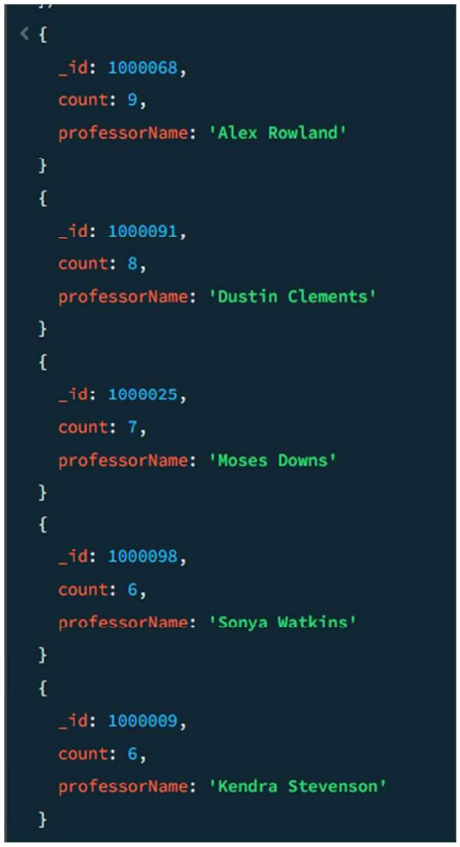
```
    _id: 0,
    SoftwareName: 1,
    Validity_Till: 1})
```

```
> db.software.find({
    Validity_Till: { $gte: ISODate("2015-01-01") }
  }, {
    _id: 0,
    SoftwareName: 1,
    Validity_Till: 1
  })
< {
    SoftwareName: 'Copyscape',
    Validity_Till: 2028-04-06T00:00:00.000Z
  }
  {
    SoftwareName: 'DupliChecker',
    Validity_Till: 2029-06-19T00:00:00.000Z
  }
  {
    SoftwareName: 'Dustball',
    Validity_Till: 2029-03-09T00:00:00.000Z
  }
  {
    SoftwareName: 'Grammarly',
    Validity_Till: 2027-12-15T00:00:00.000Z
  }
  {
    SoftwareName: 'Plagiarism Checker (Duplichecker)',
    Validity_Till: 2024-03-20T00:00:00.000Z
  }
  {
    SoftwareName: 'Plagiarism Checker (EduBirdie)',
    Validity_Till: 2029-09-07T00:00:00.000Z
  }
  {
    SoftwareName: 'Plagiarism Checker (PlagiarismCheck.org)',
    Validity_Till: 2025-12-30T00:00:00.000Z
  }
```

Query 4: Get the professor name and count of mentees (limit the answer by 5)

```
db.student.aggregate([
    {
      $group: {
      _id: "$MentorID",
      count: { $sum: 1 }
      }
    },
    {
      $sort: {
            count: -1 // Sort by count in descending order
      }
    },
    {
      $limit: 5 // Limit the result to the top 5
    },
    {
```

```
            $lookup: {
            from: "professor",
            localField: "_id",
            foreignField: "ProfID",
            as: "professor"
        }
    },
    {
        $unwind: "$professor"
    },
    {
        $project: {
        _id: 1,
        count: 1,
        professorName: "$professor.Name",
        students: 1
        }
    }
])
```



Query 5: Get the count of each report under plagiarism category

```
db.report.aggregate([
        {
        $project: {
```

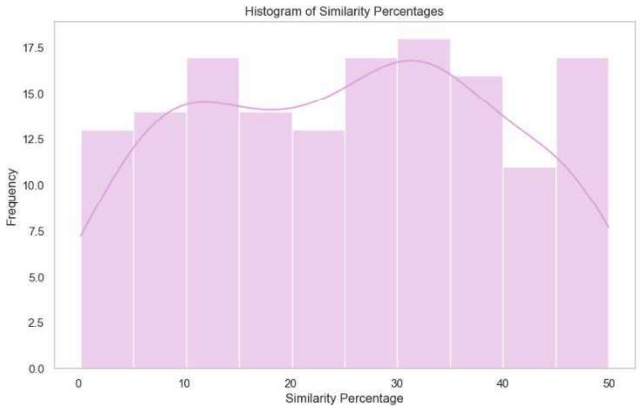
```
                _id: 0,
                SimilarityPercentage: 1,
                category: {
                        $switch: {
                                branches: [
                                        { case: { $lte: ["$SimilarityPercentage", 5] }, then: "Fair use" },
                                        { case: { $lte: ["$SimilarityPercentage", 10] }, then: "Warning" },
                                        { case: { $lte: ["$SimilarityPercentage", 20] }, then: "Serious Warning" }
                                        ],
                                default: "Blacklisted"
                                }
                        }
                }
        },
        {
                $group: {
                _id: "$category",
                ReportCount: { $sum: 1 }
        }
},
{
        $project: {
        category: "$_id",
        ReportCount: 1,
        _id: 0
}
}
])
```

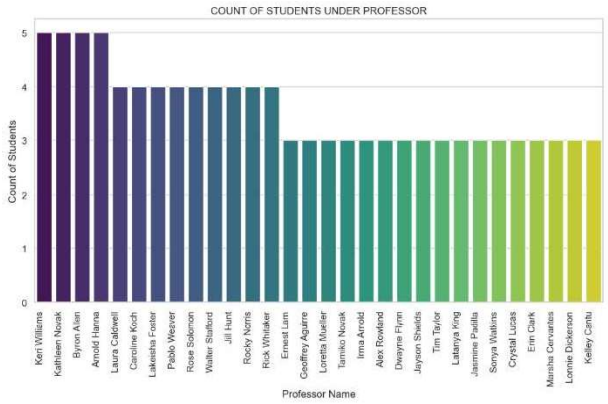
V. Database Access via Python

In the Python implementation, the MySQL connector efficiently connects to the MySQL server, enabling the execution of varied queries on student reports, project participation, professor-student relationships, and plagiarism reports. Visualization tools like Seaborn and Matplotlib translate complex data into informative charts such as histograms and pie charts. Noteworthy is the system's ability to categorize students by average plagiarism levels, offering nuanced insights for proactive measures against academic misconduct. This Python implementation underscores technical proficiency in database interaction and analytical capabilities for combating plagiarism in educational settings.

Graph 1: Distribution of the similarity %

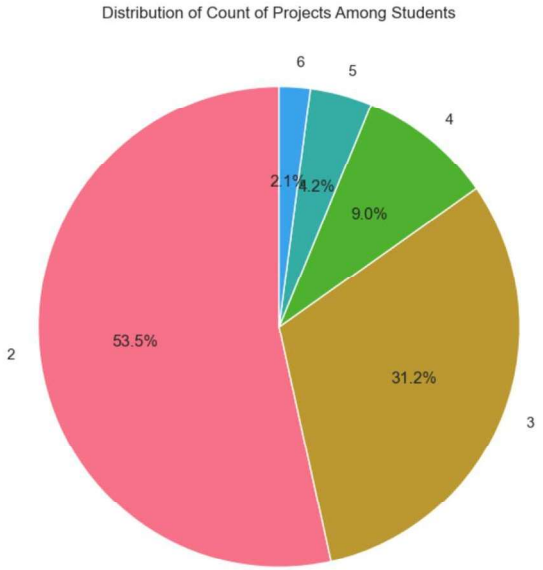
Graph 5: Categories of Plagiarism of Students ch professor

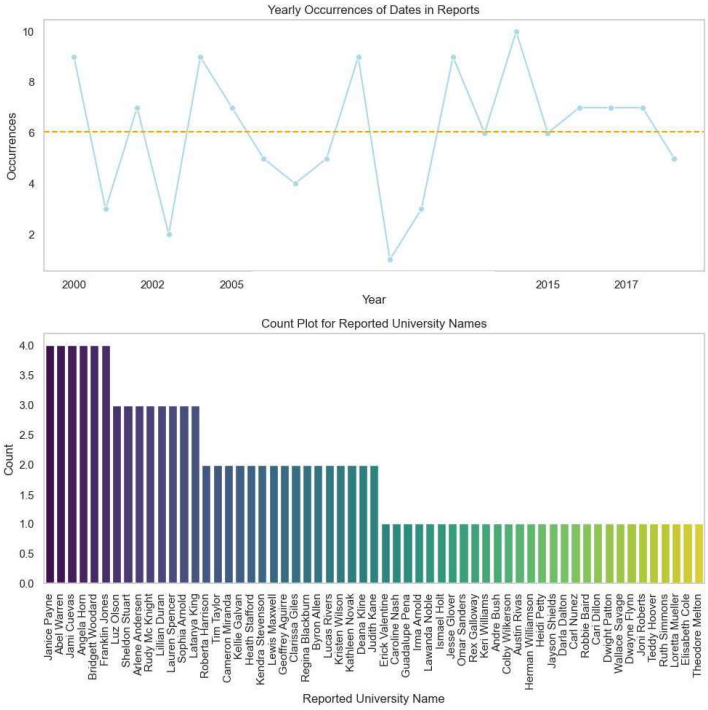


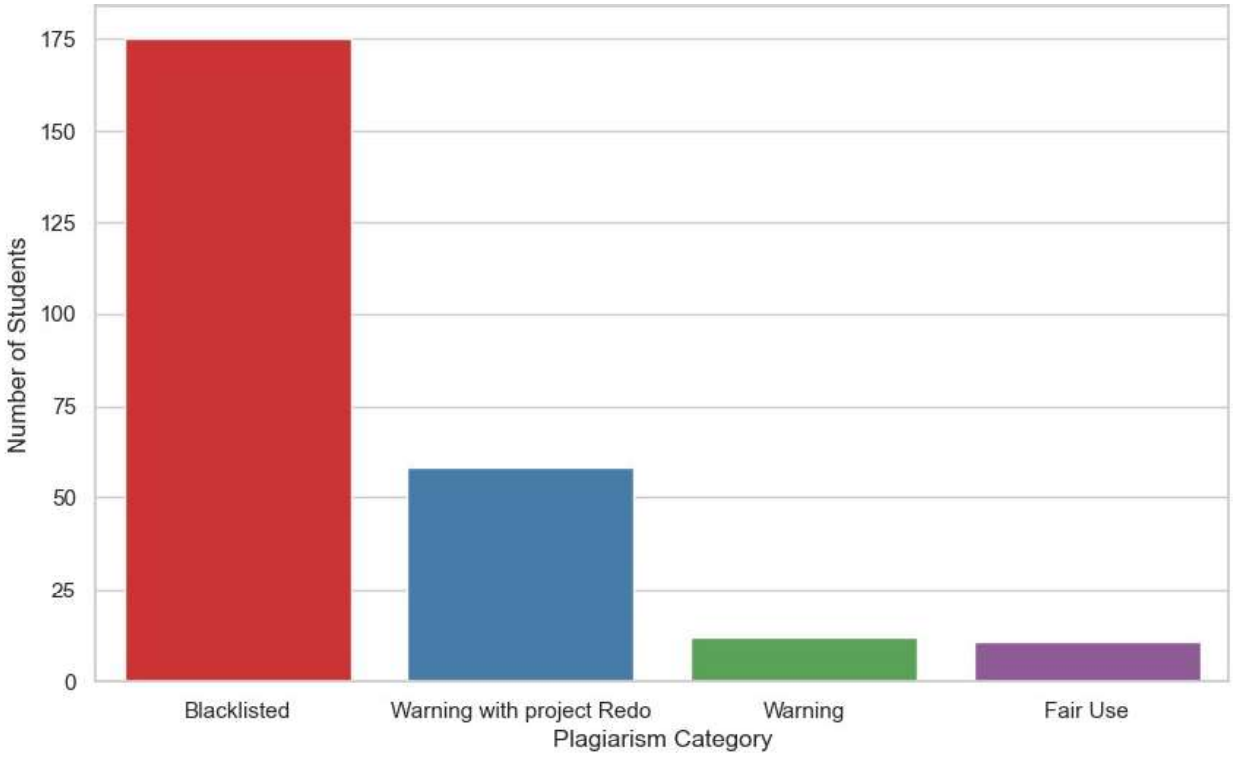


Graph 3: Visualization of number of Projects each students involved

Graph 4: Count and Distribution of Blacklisted students reported to the funding company







VI. Summary and Recommendations

In conclusion, the "Guardians of Ethics – Anti-Plagiarism Tracking Management System" has successfully addressed the pervasive issue of plagiarism in academic settings through the integration of robust relational and NoSQL databases. The introduction highlighted the necessity of fostering intellectual integrity and provided an overview of the project's objective to create a comprehensive tracking system. The subsequent implementation stages detailed the design and implementation of a relational database using MySQL, followed by the strategic inclusion of MongoDB for enhanced adaptability and unstructured data management.

The synergy between SQL and NoSQL databases has empowered the system to efficiently track and categorize plagiarism levels, providing valuable insights through sophisticated queries and data visualizations. The project's Python implementation demonstrated seamless connectivity, query execution, and data visualization, showcasing the system's analytical prowess. The strategic balance between structured and unstructured data has allowed for a flexible and scalable anti-plagiarism solution.

Moving forward, it is recommended to maintain this equilibrium, continually monitoring data growth and adapting the system to evolving requirements. Comprehensive documentation should be upheld to facilitate knowledge transfer and troubleshooting. Regular updates and security measures will ensure the system remains resilient to emerging threats. Moreover, ongoing training for the development team, combined with a robust backup and recovery strategy, will fortify the database infrastructure against unforeseen challenges. By adhering to these recommendations, this project can ensure the sustained effectiveness and adaptability of its anti-plagiarism tracking system in educational institutions.