Database Comparison Project

Project report submitted in partial fulfillment of the requirement for the degree of Bachelor of Technology

in

Computer Science and Engineering/Information Technology

By

Rohan Chaturvedi (191271)

Under the supervision of

Dr. Rajni Mohana

to

Department of Computer Science & Engineering and Information Technology

Jaypee University of Information Technology Waknaghat,

Solan-173234, Himachal Pradesh

Candidate's Declaration

I hereby declare that the work presented in this report entitled **"Database Comparison Tool"** in partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology** in **Computer Science and Engineering/Information Technology** submitted in the department of Computer Science & Engineering and Information Technology**,** Jaypee University of Information Technology Waknaghat is an authentic record of my own work carried out over a period from February 2023 to May 2023 under the supervision of **Dr. Rajni Mohana** (Associate Professor in the Department of Computer Science and Engineering).
The matter embodied in the report has not been submitted for the award of any other degree or diploma.

(Student Signature)
Rohan Chaturvedi, 191271.

This is to certify that the above statement made by the candidate is true to the best of my knowledge.

(Supervisor Signature)
Dr. Rajni Mohana
Associate Professor
Computer Science and Engineering
Dated

JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY, WAKNAGHAT

PLAGIARISM VERIFICATION REPORT

Date: ………………………….

Type of Document (Tick): **PhD Thesis** **M.Tech Dissertation/ Report** **B.Tech Project Report** **Paper**

Name: _________________________ __Department: _______________ Enrolment No ________

Contact No. ___________________________E-mail. __________________________________

Name of the Supervisor: ____

Title of the Thesis/Dissertation/Project Report/Paper (In Capital letters): ______________________

____

____

UNDERTAKING

I undertake that I am aware of the plagiarism related norms/ regulations, if I found guilty of any plagiarism and copyright violations in the above thesis/report even after award of degree, the University reserves the rights to withdraw/revoke my degree/report. Kindly allow me to avail Plagiarism verification report for the document mentioned above.

Complete Thesis/Report Pages Detail:

- Total No. of Pages =
- Total No. of Preliminary pages =
- Total No. of pages accommodate bibliography/references =

(Signature of Student)

FOR DEPARTMENT USE

We have checked the thesis/report as per norms and found **Similarity Index** at ……………….(%). Therefore, we are forwarding the complete thesis/report for final plagiarism check. The plagiarism verification report may be handed over to the candidate.

(Signature of Guide/Supervisor) **Signature of HOD**

FOR LRC USE

The above document was scanned for plagiarism check. The outcome of the same is reported below:

Copy Received on	**Excluded**	**Similarity Index (%)**	**Generated Plagiarism Report Details** (Title, Abstract & Chapters)	
	• All Preliminary Pages • Bibliography/Images/Quotes • 14 Words String		Word Counts	
Report Generated on			Character Counts	
		Submission ID	Total Pages Scanned	
			File Size	

Checked by
Name & Signature **Librarian**

……

Please send your complete thesis/report in (PDF) with Title Page, Abstract and Chapters in (Word File) through the supervisor at plagcheck.juit@gmail.com

ACKNOWLEDGEMENT

Firstly, I express my gratitude to the god who provided me with courage and fortitude to
complete the project.

I am grateful and wish my profound indebtedness to Supervisor Dr.Rajni Mohana, Associate Professor, Department of CSE Jaypee University of Information Technology, Wakhnaghat.

Deep Knowledge & keen interest of my supervisor in the field of "Data Processing" to carry out this Project. Her endless patience, scholarly guidance, continual encouragement, constant and energetic supervision, constructive criticism, valuable advice, reading many inferior drafts and correcting them at all stages have made it possible to complete this Project.

I would like to express my heartiest gratitude to Dr. Rajni Mohana, Department of CSE, for her kind help to finish my Project.

I would also generously welcome each one of those individuals who have helped me straightforwardly or in a roundabout way in making this project a success. In this unique situation, I might want to thank the various staff individuals, both educating and non-instructing, which have developed their convenient help and facilitated my undertaking.

Finally, I must acknowledge with due respect the constant support and patience of my
parents.

(Student Signature)
Project Group No.: 94
Student Name: Rohan Chaturvedi
Rollno.: 191271

Contents

S.No.	CHAPTERS	PAGE
1.	**List of Figures**	**v**
2.	**List of Graphs**	**vi**
3.	**Abstract**	**vii**
4.	**Chapter 1: Introduction**	**1**
	a. Ch 1.1: Introduction	1
	b. Ch 1.2: Problem Statement	2
	c. Ch 1.3: Objectives	4
	d. Ch 1.4: Methodology	5
	e. Ch 1.5: Organization	12
5.	**Chapter 2: Literature Survey**	**14**
6.	**Chapter 3: System Development**	**17**
	a. Ch 3.1: Related Terms	17
	i) Ch 3.1.1 Main Components of a database	19
	ii) Ch 3.1.2 Spring Framework	20
	iii) Ch 3.1.3 Connection details	21
	iv) Ch 3.1.4 Database Connections	22
	v) Ch 3.1.5 Form Controller	23
	vi) Ch 3.1.6 Pom.xml	27
	vii) Ch 3.1.7 Table Comparison Class	29
	viii) Ch 3.1.8 ResultSet	32
7.	**Chapter 4: Performance Analysis**	**37**
8.	**Conclusion**	**46**
	a. Conclusion	46
	b. Future Scope	46
9.	**References**	**48**

LIST OF FIGURES

FIGURE	PAGE NO.
Fig 1:Program Flowchart Step (1)	10
Fig 2: Program Flowchart Step (2)	10
Fig 3: Program Flowchart Step (3)	11
Fig 4: Program Flowchart Step (4)	11
Fig 5: Program Flowchart Step (5)	12
Fig 6: Code for FormData Class (2)	24
Fig 7: Code for FormData Class (2)	25
Fig 8: Connection Page UI	26
Fig 9: Code for TableComparatorClass (1)	29
Fig 10: Code for TableComparatorClass (2)	30
Fig 11: Code for TableComparator Class (3)	31
Fig 12: Code for ResultSet_Downloader Class (1)	33
Fig 13: UI for Dual Table ResultView (1)	34
Fig 14: Performing Search operations in ResultView (2)	35
Fig 15: Code for JUnits for Table Comparison Class (1)	38
Fig 16: Code for JUnits for Table Comparison Class (2)	39
Fig 17: Code for JUnits for Connection Class (1)	40
Fig 18: Code for JUnits for formData Class (1)	41
Fig 19: Code for JUnits for formData Class (2)	42
Fig 20: Console output for Table_Comparison_Test class	43
Fig 21: Console output for Connection_Test class	44
Fig 22: Console output for form_data_Test class	45

ABSTRACT

The database comparison tool presented in this project is a powerful and efficient software solution designed to facilitate the comparison and synchronization of databases. With the increasing complexity of modern database systems and the need for accurate data management, such a tool becomes essential for organizations that rely heavily on database operations.

The database comparison tool offers a comprehensive set of features and functionalities to compare the schema and data of two databases. It enables users to identify differences in table structures, column definitions, indexes, and constraints between databases. Moreover, it allows for the comparison of data records, ensuring the integrity and consistency of information across databases.

CHAPTER 1 : INTRODUCTION

1.1 Introduction

In the current age of technology, databases are becoming increasingly critical for businesses to store, manage, and retrieve data. Often, multiple environments exist for a database, such as development, testing, quality assurance (QA), pre-production, and production. The challenge for developers and DBAs is to keep the databases in these environments synchronized and up-to-date. This is where database comparison tools come into play. Database comparison tools are specialized software applications that help developers and DBAs compare databases, schemas, and data, identify discrepancies, and generate scripts to sync them. These tools provide a visual representation of the differences between databases, which makes it easy to identify changes and update them. Here are some reasons why a database comparison tool is essential in real-time development:

- Efficient Development Workflow: Developers and DBAs need to work collaboratively, and a database comparison tool helps them work efficiently. The tool identifies the differences between two databases and generates scripts to make the changes. This eliminates the need to manually compare and identify changes, which can be time-consuming and prone to errors.
- Avoiding Data Loss: In the database comparison process, the tool identifies the differences between two databases, which include changes in the schema or data. Without a database comparison tool, developers and DBAs may miss a change, leading to data loss. The tool ensures that all changes are identified, and nothing is lost.

- Better Quality Assurance: Database comparison tools are invaluable for quality assurance (QA) teams. These tools allow QA teams to test changes and new features in a separate environment, such as a staging or pre-production environment, before deploying to production. This ensures that changes are thoroughly tested and reduces the risk of bugs and other issues in the production environment.
- Compliance and Security: Database comparison tools can help ensure compliance with regulations and standards such as HIPAA and GDPR. The tool can identify changes to the database that may affect compliance and generate scripts to bring the database into compliance. Additionally, the tool can help ensure the security of sensitive data by identifying differences in access control and other security settings.
- Scalability: As databases grow, it can become challenging to keep track of changes and maintain synchronization between environments. A database comparison tool makes it easy to manage and synchronize databases of any size, whether it is a small database or a large enterprise-level database.

Hence, a database comparison tool is essential in real-time development because it helps developers and DBAs work efficiently, avoid data loss, perform better quality assurance, ensure compliance and security, and scale with ease. With the growth of databases and the need to maintain multiple environments, database comparison tools are becoming more critical than ever. They streamline the development process, reduce errors, and increase productivity, making them a must-have for any development team.

1.2 Problem Statement

With the increasing complexity and size of databases in modern software systems, it has become necessary to have a tool that can help developers and database administrators to identify differences between different database versions and environments. This tool can be used to compare the databases across different servers like production, QA, pre-production, and more. In a typical software development process, there are multiple environments where databases are maintained, such as development, testing, staging, and production. The data in these databases should be identical or at least consistent across all the environments. However, due to various reasons such as human errors, differences in software versions, hardware or network differences, database configuration differences, etc., there may be variations between the databases on different environments.

In such scenarios, identifying the differences between the databases can be a time-consuming and error-prone task, as manual comparison of database schemas and data can be challenging and prone to errors. It is essential to identify and resolve these differences between the databases as early as possible to ensure that the software is running as expected in all the environments.

A database comparison tool provides a solution to this problem by automating the comparison of databases, which helps to save time, reduce errors, and ensure accuracy. With a database comparison tool, the user can compare the schema of two databases, including tables, columns, indexes, triggers, and other database objects, to identify differences between them. Additionally, the tool can also compare the data in tables and identify differences in the data. These tools usually have a user-friendly interface and allow the user to select the databases to be compared and customize the comparison settings. They can also provide features such as generating reports, scripting, and synchronization of databases. They can also help in identifying issues such as performance bottlenecks, optimization opportunities, and security vulnerabilities.

In conclusion, a database comparison tool is essential for any software development project, as it helps in detecting differences between the databases and ensuring that the software is running as expected across all the environments. The tool can save time, reduce errors, and ensure accuracy in the comparison process. There are many database comparison tools available in the market, both free and commercial, that cater to different requirements and preferences. It is essential to select the right tool that meets the project's needs and provides accurate and reliable results.

1.3 Objectives

The objective of a database comparison tool is to help developers, database administrators, and other stakeholders in software development to easily compare and synchronize the data and schema of databases. Here are some of the key objectives of such a tool:

- Easy Comparison: The tool should provide an easy and efficient way to compare databases, including schemas and data. This can be done using a graphical user interface (GUI) or command-line interface (CLI) that allows users to quickly see the differences between the databases.

- Automate Comparison: The tool should be able to automate the comparison process so that it can be performed on a regular basis or as part of a continuous integration/continuous delivery (CI/CD) pipeline. This helps ensure that databases are always in sync and that any changes made to one database are reflected in the others.

- Support Multiple Databases: The tool should be able to compare databases of different types, including Oracle, MySQL, SQL Server, and others. It should also be able to handle different versions of the same database.

- Support Data Migration: The tool should be able to migrate data from one database to another, making it easy to move data between different environments. This is particularly useful for developers who need to move data from a production database to a development or test environment.

- Support Schema Comparison: The tool should be able to compare the schema of different databases, including tables, views, stored procedures, and other database objects. It should also be able to detect changes in the schema and alert users to any potential issues.

- Ensure Data Integrity: The tool should be able to detect and report any data inconsistencies between the databases being compared. This helps ensure that data is accurate and that there are no data integrity issues that could affect the functioning of the software application.

- Provide Reports: The tool should provide detailed reports on the differences between the databases being compared. These reports should be easy to understand and should provide enough detail to allow developers to make informed decisions about how to synchronize the databases.

- Provide Security: The tool should provide adequate security measures to protect the sensitive data stored in the databases being compared. This might include data encryption, access controls, and other security features.

1.4 Methodology

1.4.1 Spring Framework

Spring is an open-source framework for developing Java applications. It provides a comprehensive set of features that make it easy to develop modular, scalable, and secure applications. Spring is one of the most popular Java frameworks and is used by a wide range of organizations, including Google, Amazon, and Netflix. pring is an open-source framework for developing Java applications. It provides a comprehensive set of features that make it easy to develop modular, scalable, and secure applications. Spring is one of the most popular Java frameworks and is used by a wide range of organizations, including Google, Amazon, and Netflix. Spring is a lightweight framework, which means that it does not add a lot of overhead to your applications. It is also very flexible, so you can use it to develop a wide variety of applications. Spring is also very well-documented, so it is easy to learn and use.

Some of the key features of Spring include:

- Dependency injection: Dependency injection is a technique that allows you to decouple your code from its dependencies. This makes your code more modular and easier to test.
- IoC container: The IoC container is a central part of Spring. It is responsible for managing the dependencies of your application.
- AOP: AOP is a technique that allows you to add cross-cutting concerns to your code. This makes your code more reusable and easier to maintain.
- ORM: Spring provides a number of ORM (object-relational mapping) frameworks that make it easy to interact with databases.
- MVC: Spring supports the MVC (model-view-controller) architectural pattern. This makes it easy to develop web applications.
- Security: Spring provides a number of features that make it easy to secure your applications.
- Testing: Spring provides a number of features that make it easy to test your applications.

Spring is a powerful and versatile framework that can be used to develop a wide variety of Java applications. If you are looking for a framework that can help you develop modular, scalable, and secure applications, then Spring is a good choice.

Here are some additional benefits of using Spring:

- Spring is a well-established framework with a large community of users and developers. This means that there are a lot of resources available to help you learn and use Spring.
- Spring is a mature framework that has been around for many years. This means that it has been well-tested and is reliable.

1.4.2 Spring Boot

Spring Boot is an open-source framework that makes it easy to create stand-alone, production-grade Spring applications. Spring Boot takes care of a lot of the boilerplate code that is typically required to get a Spring application up and running, so you can focus on writing your application logic.
Sure. Here is a detailed explanation of Spring Boot for a project report:

Spring Boot is an open-source framework that makes it easy to create stand-alone, production-grade Spring applications. Spring Boot takes care of a lot of the boilerplate code that is typically required to get a Spring application up and running, so you can focus on writing your application logic.

Spring Boot is built on top of the Spring Framework, but it provides a number of features that make it easier to use. Some of the key features of Spring Boot include:

- Auto-configuration: Spring Boot automatically configures many of the common Spring features, so you don't have to do it yourself.
- Starters: Spring Boot provides a number of starters that make it easy to add common Spring features to your application.
- Convention over configuration: Spring Boot uses convention over configuration, which means that it makes assumptions about how your

application is configured. This can make it easier to get started with Spring Boot.

- Production-ready features: Spring Boot includes a number of production-ready features, such as embedded web servers, security, and metrics.

Spring Boot is a great choice for developers who want to create Spring applications quickly and easily. It is also a good choice for developers who want to create production-ready applications.

1.4.3 MVC Architecture

MVC architecture is a software design pattern that separates the presentation (view), the data (model), and the business logic (controller). This separation of concerns makes it easier to develop, test, and maintain applications. MVC is a popular architecture for developing web applications and is supported by many popular frameworks, including Spring.

1.4.4 Maven Project

Maven is a build automation tool that helps you automate the process of building, testing, and deploying Java applications. Maven provides a standard directory structure for Java projects and a set of conventions for naming files and directories. This makes it easy to build and deploy Java applications with Maven.
Sure. Here is a detailed explanation of Maven for a project report:

Maven is a build automation tool that helps you automate the process of building, testing, and deploying Java applications. Maven provides a standard directory structure for Java projects and a set of conventions for naming files and directories. This makes it easy to build and deploy Java applications with Maven.

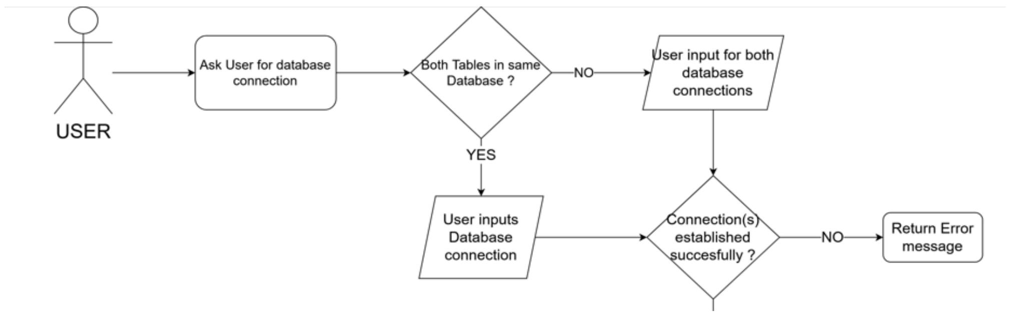
Maven uses a project object model (POM) to describe your project. The POM is an XML file that specifies the dependencies of your project, the build lifecycle, and other information. Maven uses the POM to build your project and to deploy it to a repository.

Maven has a number of advantages over other build automation tools, including:

- It is a standard tool. Maven is a standard tool that is used by a large number of Java developers. This means that there are a lot of resources available to help you learn and use Maven.
- It is easy to use. Maven is easy to use, even for beginners. The POM makes it easy to specify the dependencies of your project and the build lifecycle.
- It is powerful. Maven is a powerful tool that can be used to build and deploy Java applications of any size.
- It is scalable. Maven can be used to build and deploy Java applications of any size.
- It is secure. Maven uses a number of security features to protect your projects from security threats.

1.4.5 FlowChart

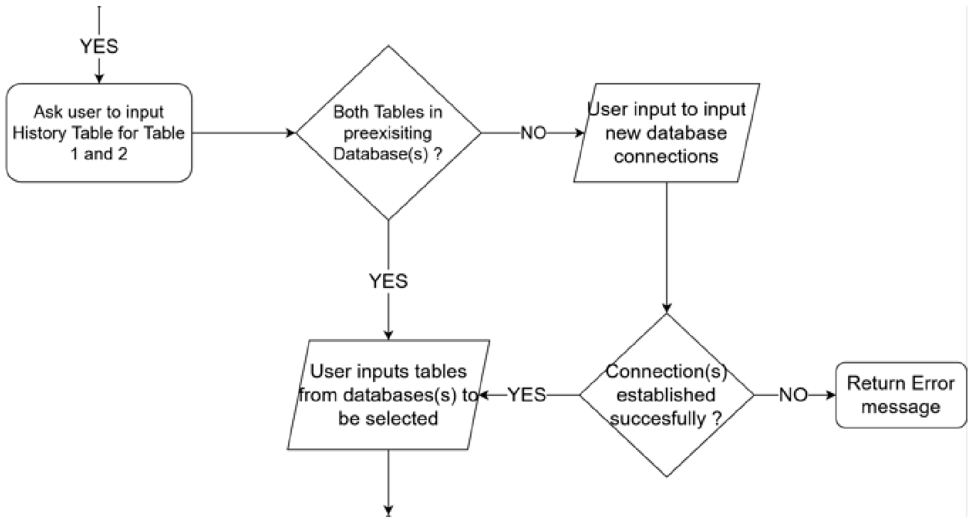
1. The program initiated by the user starting up the server.



USER
Ask User for database connection
Both Tables in same Database ?
NO
User input for both database connections
YES
User inputs Database connection
Connection(s) established succesfully ?
NO
Return Error message

FIG 1: PROGRAM FLOWCHART STEP (1)

To start comparing tables, the user is required to input a database connection. If the user is comparing two tables, they must input the database connection for one database if both tables are in the same database. If the tables belong to different databases, then the user needs to input two database connections. If the user is working with a single table and its history table for previous record values, then they input the database connection for the table, assuming both the table and the history table exist in the same database.



YES
Ask user to input History Table for Table 1 and 2
Both Tables in preexisiting Database(s) ?
NO
User input to input new database connections
YES
User inputs tables from databases(s) to be selected
YES
Connection(s) established succesfully ?
NO
Return Error message

FIG 2: PROGRAM FLOWCHART STEP (2)

2. The program checks whether the database connections were established successfully or not. If the connections failed, the user receives an error message indicating invalid details.

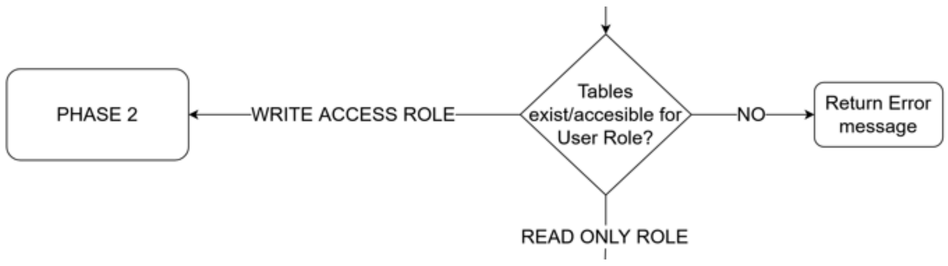


FIG 3: PROGRAM FLOWCHART STEP (3)

3. If the tables selected by the user exist and are accessible by their user role, the tables' descriptions, columns, and metadata are displayed in the program's user interface (UI). If the tables do not exist or the user does not have read/write access to them, an error message is displayed. The tables are shown in the UI in a split window, with each table in a separate window (or history table) with command and operation buttons at the top of the window.

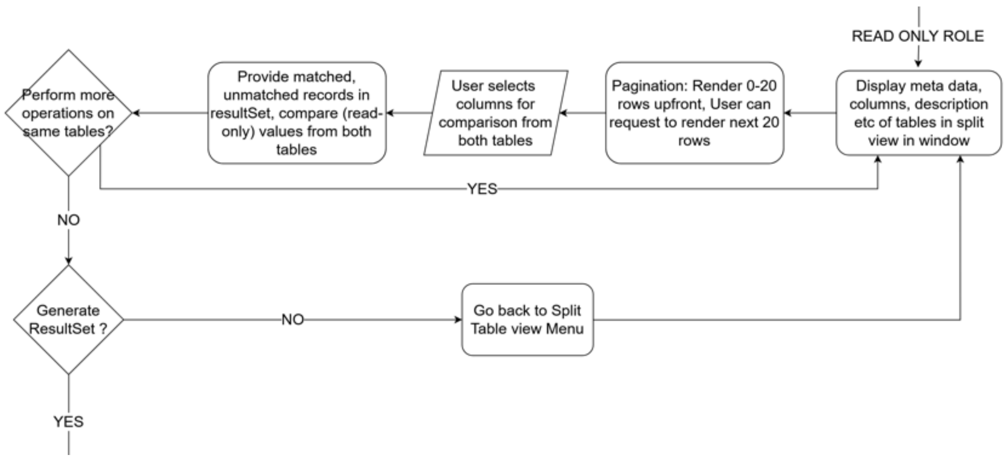

READ ONLY ROLE
Display meta data, columns, description etc of tables in split view in window
Pagination: Render 0-20 rows upfront, User can request to render next 20 rows
User selects columns for comparison from both tables
Provide matched, unmatched records in resultSet, compare (read-only) values from both tables
Perform more operations on same tables?
YES
NO
Generate ResultSet ?
NO
Go back to Split Table view Menu
YES

FIG 4: PROGRAM FLOWCHART STEP (4)

4. The user selects columns from the list of columns displayed on both tables in the UI. The program then compares column properties, values, and data types, returning matched and unmatched records from the selected columns in the UI.
5. Users can go back to select other columns from the selected tables and apply operations. The user can also terminate the operations cycle and

see the changes made in the selected tables by a previously saved commit from the tables' history tables.

6. Users can proceed to download the result set after completing all the necessary operations.

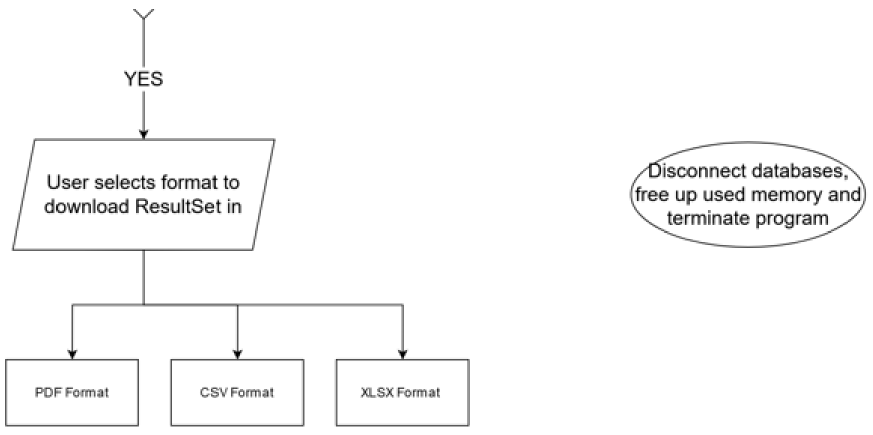

YES
User selects format to download ResultSet in
PDF Format
CSV Format
XLSX Format
Disconnect databases, free up used memory and terminate program

FIG 5: PROGRAM FLOWCHART STEP (5)

7. After the user closes/ logs out of the site, the server will automatically deallocate, free up used memory and terminate the program instance.

1.5 Organization

Chapter 1: Introduction

Various aspects of the project are dealt with in this section, which includes a synopsis of the assignment, the approach that was used, an explanation of the issue discussed by the project, as well as the goal of the project.

Chapter 2: Literature Survey

This project's literature review part discusses the evaluated resources as well as the topics that have been investigated and recognized.

Chapter 3: System Design and Development

Within this section, we are going to look over project analysis and system design implementation. We will describe the algorithms employed and share project snapshots.

Chapter 4: Experiment and Result Analysis

This portion of the project shows the results of the analysis by displaying and comparing the findings in snapshots. It also has the ability to show multiple outputs. The part also discusses how the outcomes were acquired and what they indicate for the project's achievement.

Chapter 5: Conclusion

This component of the study ends with the present project and discusses potential additional study and development opportunities.

CHAPTER 2 : LITERATURE SURVEY

The numerous advancements in the field of networking and socket programmes were covered in this part. The effectiveness of the system and socket programming methodologies, as well as the outcomes of their testing on the specified architecture.

2.1 Research Material / Books :

- The book "Database Design for Mere Mortals: A Hands-On Guide to Relational Database Design" by Michael J. Hernandez can provide valuable insights for the development of a database comparison tool. It covers the fundamental concepts and techniques of database design that are essential for understanding the underlying principles behind the comparison of database entries. By studying this book, we can gain a better understanding of the various techniques that can be used for comparing different database schemas and entries, which can be useful in the design and implementation of our comparison tool.

- The textbook "Database Systems: Design, Implementation, and Management" by Carlos Coronel, Steven Morris, and Peter Rob can be a useful reference for developing our database comparison tool. It provides a solid foundation for understanding database systems and their management, which can help in the development of our tool. This textbook covers various aspects of database systems, including design, implementation, and management, which can provide useful insights for developing a comparison tool that can analyze and compare large datasets.

- The book "Database Modeling and Design: Logical Design" by Toby J. Teorey, Sam S. Lightstone, and Tom Nadeau can provide valuable insights for designing a comparison tool that can analyze and compare different database schemas. It offers a detailed guide to the logical design of databases, which can be useful in the development of our tool. By studying this book, we can gain a better understanding of the various techniques and strategies that can be used to compare different database schemas and entries.

- The research journal "ACM Transactions on Database Systems" can be a useful resource for the development of our database comparison tool. It publishes research papers on various topics related to database systems, including data modeling, schema design, and database management. By studying the papers published in this journal, we can gain a better understanding of the latest trends and techniques in the field of database systems, which can be useful in the development of our tool.

- The "IEEE Transactions on Knowledge and Data Engineering" is a research journal that covers various aspects of knowledge and data engineering, including database design, data mining, and information retrieval. The papers published in this journal can provide useful insights and ideas for the development of a comparison tool that can analyze and compare large datasets. By studying the research papers published in this journal, we can gain a better understanding of the various techniques and strategies that can be used to analyze and compare different databases.

- The book "Advances in Database Technology - EDBT 2021" is a collection of papers presented at the 24th International Conference on Extending Database Technology. It covers recent advancements in database technology, including data modeling, query processing, and database management. By studying this book, we can gain a better understanding of the latest trends and techniques in the field of database technology, which can be useful in the development of our comparison tool.

- The "Journal of Database Management" is a research journal that covers various aspects of database management, including data modeling, database design, and database security. By studying the papers published in this journal, we can gain a better understanding of the various techniques and strategies that can be used to compare and analyze different databases. The insights gained from this journal can be useful in the development of our comparison tool.

- The "Data Mining and Knowledge Discovery" journal covers various aspects of data mining and knowledge discovery, including pattern recognition, clustering, and data visualization. The papers published in this journal can be relevant to the development of a tool that can analyze and compare large datasets. By studying the research papers published in this journal, we can gain a better understanding of the various techniques and strategies that can be used to analyze and compare different databases, which can be useful in the development of our comparison tool.

CHAPTER 3 : SYSTEM DESIGN & DEVELOPMENT

3.1 System Design

3.2.1 Main components of Database tool

The system design for the database comparison tool involves several components that work together to enable users to compare database entries. The following sections provide an overview of each component and its functionality.

1. User Interface (UI):
The user interface is the front-end component of the database comparison tool, which allows users to interact with the system. It consists of various windows, buttons, and forms that enable users to input the necessary information and initiate database comparison operations. The UI should be designed to be user-friendly and intuitive, so that users can easily navigate through the system and perform tasks.

2. Database Connection:
The tool must be able to establish a connection to the database(s) that users want to compare. Depending on the comparison scenario, the user may be required to input the connection details for one or more databases. The tool should be designed to verify the validity of the connection details and display an error message if the connection cannot be established.

3. Table Selection:
Once a database connection is established, the user can select the tables that they want to compare. The tool should be able to retrieve the metadata for

each table and display it on the UI, including the column names, data types, and descriptions. The user should be able to select the columns that they want to compare.

4. Comparison Operations:
The tool should be able to compare the selected columns of the two tables and identify any matched or unmatched records. The comparison algorithm should be designed to handle different data types and formats, and should be able to perform various types of comparison operations, such as exact match, partial match, and fuzzy match.

5. Operation Buttons:
The UI should include buttons that allow users to perform various operations on the selected tables, such as adding new records, updating existing records, and deleting records. These buttons should be disabled if the user does not have the necessary permissions to perform the operation.

6. History Table:
If the user is comparing a table with its history table for previous record values, the user must input the database connection for the history table. The tool should be able to retrieve the data from the history table and display it on the UI. The user should be able to commit the changes made to the table to the history table.

7. Result Set:
The tool should be able to generate a result set based on the comparison operation performed by the user. The result set should display the matched and unmatched records in a user-friendly format, such as a table or a chart. The user should be able to export the result set to various formats, such as CSV or Excel.

3.1.2 Spring Framework Architecture

The Spring Framework architecture is designed to be modular and extensible, allowing developers to choose the components that they need and leaving out the ones that they do not need. The Spring Framework consists of several modules that can be used independently or in combination with other modules. The core module of the Spring Framework is the Spring Container, which is responsible for managing the lifecycle of Java objects. The container is responsible for creating, initializing, and managing the objects, and it provides a layer of abstraction between the application and the underlying Java classes.

The Spring Framework also includes several other modules, such as Spring MVC, which is used for building web applications, and Spring Data, which provides support for data access and persistence. Spring Security is another module that provides authentication and authorization functionality for securing web applications.

The key components of the Spring Framework include the following:

1. Inversion of Control (IoC) Container: The IoC container is responsible for managing the lifecycle of Java objects and providing dependency injection. The container creates and manages the objects, and it injects the required dependencies into the objects at runtime.

2. Spring MVC: Spring MVC is a module of the Spring Framework that is used for building web applications. It provides a Model-View-Controller (MVC) architecture for separating the presentation layer from the business logic and data access layer.

3. Spring Data: Spring Data is a module of the Spring Framework that provides support for data access and persistence. It supports several

data stores, such as relational databases, NoSQL databases, and cloud storage services.

4. Spring Security: Spring Security is a module of the Spring Framework that provides authentication and authorization functionality for securing web applications. It supports several authentication mechanisms, such as form-based authentication, OAuth, and OpenID Connect. These security features are great but we would not be using this database tool as an open source available to public tool hence a lot of security comes from it being closed/ proprietary software.

How Spring Framework Works:

The Spring Framework works by providing a set of pre-built components and modules that developers can use to build enterprise applications. The framework provides a layer of abstraction between the application code and the underlying Java classes, making it easier to manage and maintain the code. The IoC container is responsible for creating and managing the objects, and it injects the required dependencies into the objects at runtime. This allows developers to focus on the business logic of the application rather than worrying about managing the object lifecycle and dependencies.

The AOP framework is used for modularizing cross-cutting concerns, such as logging and transaction management. AOP allows developers to separate the concerns of an application into different modules, making it easier to manage and maintain the code. The Spring MVC module provides a Model-View-Controller (MVC) architecture for separating the presentation layer from the business logic and data access layer. This makes it easier to build web applications that are easy to manage and maintain.

Using Spring MVC, we are able to divide our web application into three distinct layers, which are the model, view, and controller. The model represents the data and the business logic of the application, the view represents the user interface, and the controller acts as the mediator between the model and the view. This separation of concerns makes our code more modular, maintainable, and easier to test.

3.4 Connection details Front to Back-End: Form Data Class

is a Java class named "FormData" which is used to store and communicate the connection data for different MySQL-connection classes. The class has private member variables such as "serverUrl", "database1", "database2", "portNumber", "username", "password", and "tableName" that are used to store the details for establishing the connection with the MySQL database.

The class provides two constructors, including a default constructor and a parameterized constructor, which initialize the private member variables of the class. The parameterized constructor takes the values for the "serverUrl", "database1", "database2", "portNumber", "username", "password", and "tableName" variables and initializes the corresponding private member variables of the class with these values. The class provides public getter and setter methods for all the private member variables, which allow accessing and modifying the values of these variables. For example, the "getServerUrl()" method returns the value of the "serverUrl" private member variable, and the "setServerUrl(String serverUrl)" method sets the value of the "serverUrl" private member variable to the passed parameter. The "isValid()" method is used to check the validity of the connection details. This method checks if all the private member variables of the class are initialized and have non-empty values. If any of the values are null or empty, the method returns false, otherwise, it returns true.

In conclusion, the "FormData" class is a vital component of the project that stores and communicates the MySQL database connection details for various connection classes. This class is designed to ensure that all the required details for establishing a connection with the database are available, and the provided values are valid before proceeding with the connection process.

3.5 Testing Database Connections

Most of our data is stored and handled by Oracle Db in our company hence, our end specification will be tailored to work best with Oracle Db servers and Databases. During Development we used MySQL Database Servers due to their Open source nature and minimal access needed. This class MysqlConnection.java deals with connection with two databases and other methods. using JDBC (Java Database Connectivity) to connect with a MySQL database server. The code is designed to establish a connection with MySQL using the provided database URL, username, and password. It also has methods to check if the provided databases exist and if the provided table is accessible in both databases. The code uses the Apache logging library to log events and to generate logs. In addition, the code provides a method to retrieve data from the provided table in both databases and store them in an ArrayList.

Code Structure- The code consists of a single class named MysqlConnection. It includes several import statements to import required classes and libraries. The code has a private constructor and a private instance of the MysqlConnection class. It has a Connection object to establish the connection with the MySQL database. The connection object is initialized using the provided database URL, username, and password. The code uses the Singleton design pattern to ensure that only one instance of the MysqlConnection class can be created at a time.

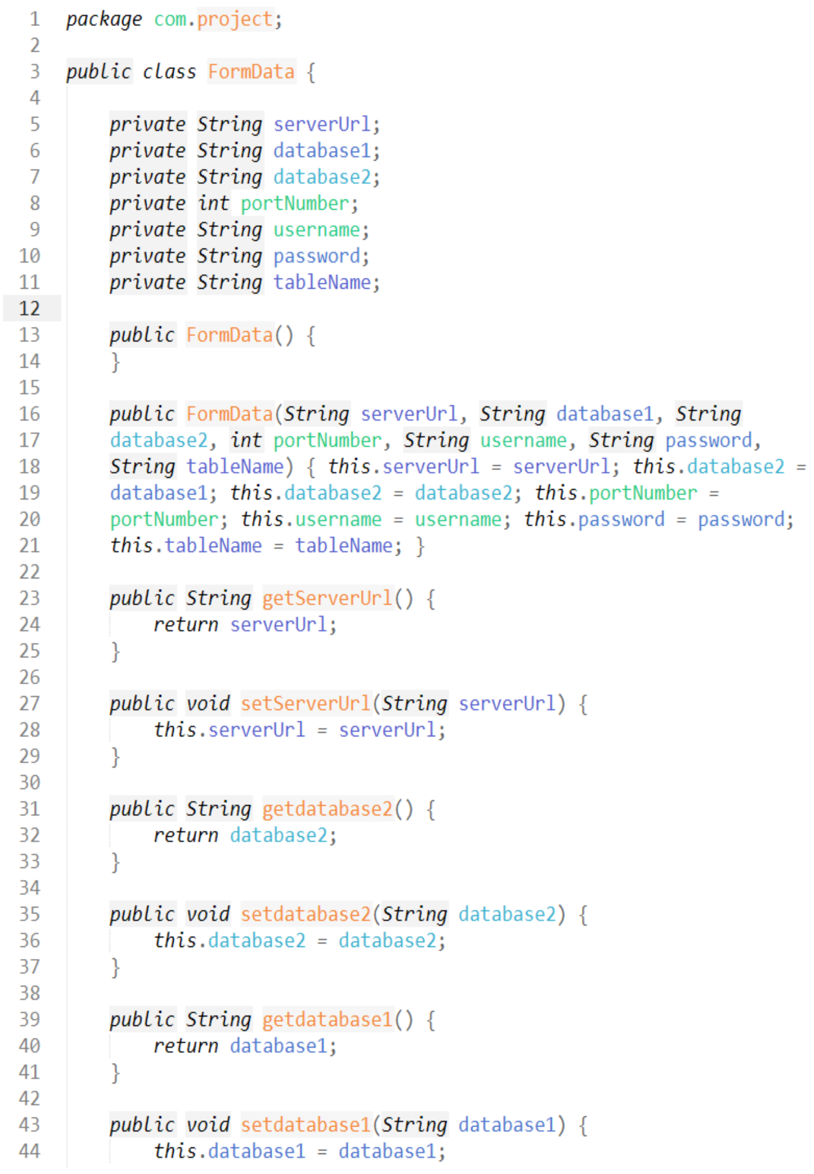
Methods-The code has several methods to perform database operations. The getConnection method returns the established connection with the MySQL database. The closeConnection method is used to close the connection to the database. The checkDatabases method is used to check if the provided databases exist and if the provided table is accessible in both databases. The method takes three parameters: two database names and the table name. The method returns true if both databases exist and the table is accessible in both databases; otherwise, it returns false.

The getTablesFromDatabases method is used to retrieve data from the provided table in both databases. The method takes three parameters: two database names and the table name. The method returns a List of Lists of Lists of Strings, which represents the retrieved data. The method retrieves data from both databases and stores it in a List of Lists of Strings. The first List represents the table from the first database, and the second List represents the table from the second database. Each table is represented by a List of Lists of Strings, where each List represents a row in the table, and each String represents a value in the row.

Logging- The code uses the Apache logging library to log events and to generate logs. The library is used to create a logger object named logger, which is used to log events. The logger object logs events of different levels, such as info, debug, error, and warning. The logs generated by the logger object can be stored in a file or displayed on the console.

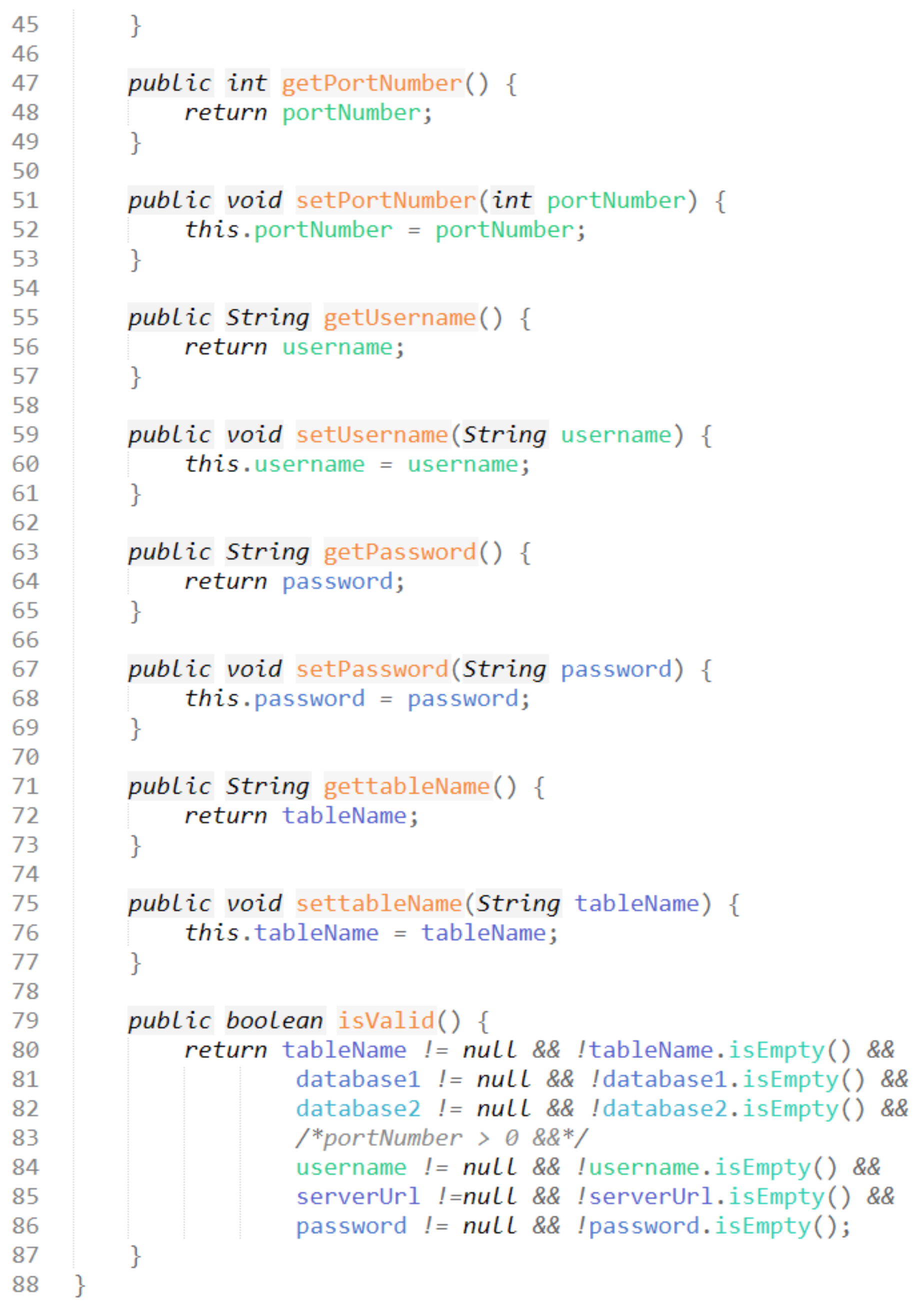
3.4 Form Controller

This Controller class in spring is the file that acts as a controller for a web application. The controller receives HTTP requests and processes them by executing the appropriate business logic. In this case, the controller establishes a connection with a MySQL database and performs some operations on it and creates object which is used for storing and communication of connection data to various SQLDB-connection classes



```java
package com.project;

public class FormData {

    private String serverUrl;
    private String database1;
    private String database2;
    private int portNumber;
    private String username;
    private String password;
    private String tableName;

    public FormData() {
    }

    public FormData(String serverUrl, String database1, String
    database2, int portNumber, String username, String password,
    String tableName) { this.serverUrl = serverUrl; this.database2 =
    database1; this.database2 = database2; this.portNumber =
    portNumber; this.username = username; this.password = password;
    this.tableName = tableName; }

    public String getServerUrl() {
        return serverUrl;
    }

    public void setServerUrl(String serverUrl) {
        this.serverUrl = serverUrl;
    }

    public String getdatabase2() {
        return database2;
    }

    public void setdatabase2(String database2) {
        this.database2 = database2;
    }

    public String getdatabase1() {
        return database1;
    }

    public void setdatabase1(String database1) {
        this.database1 = database1;
```

FIG 6: CODE FOR FORM_DATA CLASS (1)



```java
    }

    public int getPortNumber() {
        return portNumber;
    }

    public void setPortNumber(int portNumber) {
        this.portNumber = portNumber;
    }

    public String getUsername() {
        return username;
    }

    public void setUsername(String username) {
        this.username = username;
    }

    public String getPassword() {
        return password;
    }

    public void setPassword(String password) {
        this.password = password;
    }

    public String gettableName() {
        return tableName;
    }

    public void settableName(String tableName) {
        this.tableName = tableName;
    }

    public boolean isValid() {
        return tableName != null && !tableName.isEmpty() &&
                database1 != null && !database1.isEmpty() &&
                database2 != null && !database2.isEmpty() &&
                /*portNumber > 0 &&*/
                username != null && !username.isEmpty() &&
                serverUrl !=null && !serverUrl.isEmpty() &&
                password != null && !password.isEmpty();
    }
}
```

FIG 7: CODE FOR FORM_DATA CLASS (2)

The code uses the Spring Framework's annotations to map the HTTP requests to the appropriate methods in the controller. It also uses a custom class to establish a connection with the database. The class starts with a package declaration followed by a list of import statements. The import statements bring in the required classes from various packages used in the code. The classes imported include FormData, Connection, DriverManager, ResultSet, Statement, List, HttpServletRequest, HttpServletResponse, HttpServlet, Controller, Model, ModelAttribute, RedirectAttributes, and RequestMethod. The class definition starts with an annotation, @Controller, which is a Spring Framework annotation indicating that the class is a controller. The class extends HttpServlet, indicating that it is a servlet.

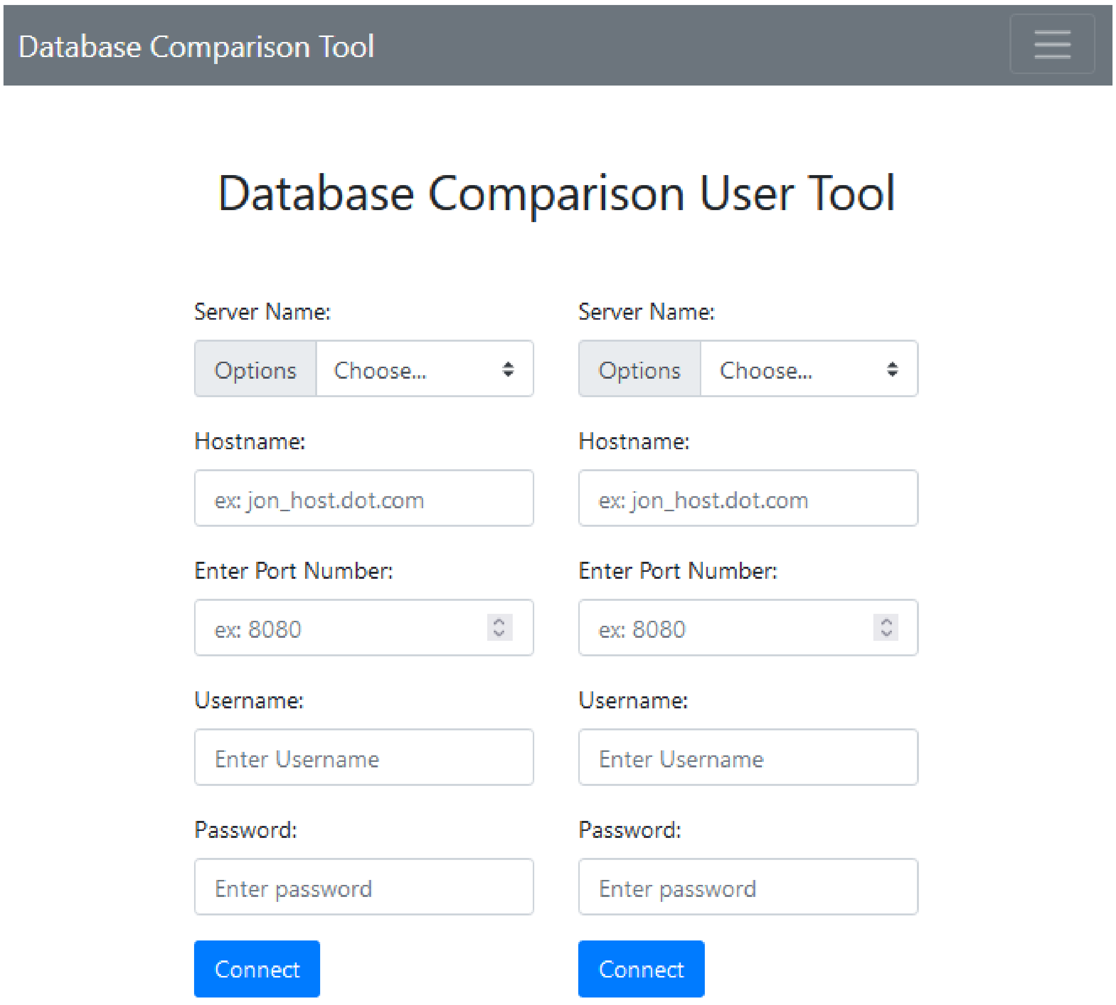


FIG 8: CONNECTION PAGE UI

The class has two methods with @RequestMapping annotations. The first method, serverConnect(), handles the HTTP GET requests for the root path and /connect path. This method creates an object of the FormData class, provides it to the model, and returns the name of the view file. The second method, submitForm(), handles the HTTP POST requests for the /submitForm path. This method takes the form data as an input and attempts to establish a connection with a MySQL database using the provided credentials. It then checks whether the specified databases contain the specified table. If both databases are accessible and contain the table, the method adds a success message to the redirectAttributes and returns the name of the view file. If there is an error, it adds an error message to the redirectAttributes and redirects the user back to the /connect path. The method uses a custom class, MysqlConnection, to establish a connection with the MySQL database. It uses the getInstance() method of the MysqlConnection class to create a new connection object. The method then calls the checkDatabases() method of the connection object to check whether both databases exist and contain the specified table.

If the checkDatabases() method returns true, the method calls the getTablesFromDatabases() method of the connection object to get a list of tables that match the specified table name in both databases. It then adds this list to the request object as an attribute with the key "tables". Finally, the method returns the name of the view file, which is confirmForm.jsp, if there is no error, or redirects the user back to the /connect path if there is an error.

3.6 Pom.xml (Maven)

The code snippet above represents an XML file that is commonly used in Java projects that use the Maven build automation tool. The XML file contains a set of configurations that define the project's dependencies and build settings.
This XML file contains the necessary configuration for a Java web application that uses several popular libraries for web development and logging. It defines the dependencies required by the project, as well as the build process for generating the final artifact. With this configuration, developers can easily

build and deploy their Java web application without worrying about manually configuring the dependencies and build process.

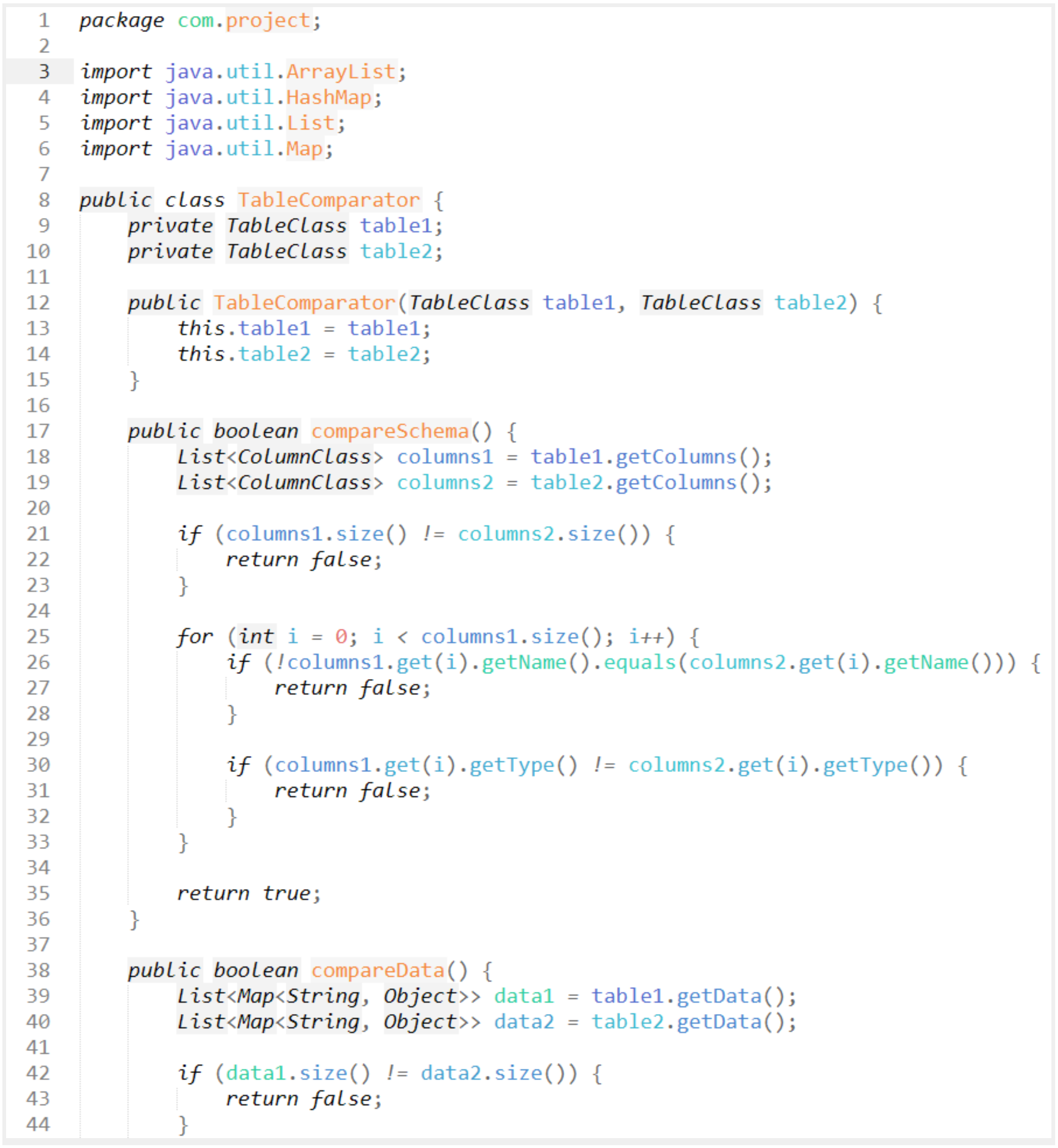
At the top level, the file declares the project namespace and version using the XML namespaces "http://maven.apache.org/POM/4.0.0" and the other as "http://www.w3.org/2001/XMLSchema-instance", respectively. The project's information, such as the group ID, artifact ID, and version, is then specified within the <modelVersion>, <groupId>, <artifactId>, and <version> tags. The <packaging> tag specifies the type of artifact that is being produced, in this case, a "war" (Web Archive) file. The <name> tag specifies the name of the project, and the <url> tag specifies the URL of the project's homepage.

The <dependencies> section lists all the external libraries that are required by the project. The dependencies are declared using the <dependency> tag, which contains the group ID, artifact ID, and version of the library. In this particular case, the project depends on several libraries, including JUnit, Spring Web MVC, and JSTL, which are all used for web development, as well as Log4j, which is a logging framework. The <scope> tag specifies the context in which the dependency is used. For instance, the "test" scope indicates that JUnit is only required for running the project's unit tests.

The <build> section specifies the configuration for the project's build process. The <finalName> tag sets the name of the output file that is generated by the build process. The <plugins> section contains a set of Maven plugins that are used during the build process. The first plugin, "spring-boot-maven-plugin", is used to generate an executable JAR file for the project. The second plugin, "maven-war-plugin", is used to generate a "war" file, which is a format commonly used for deploying web applications on a Java application server. The third plugin, "maven-compiler-plugin", is used to configure the compiler settings for the project. The <source> and <target> tags specify the Java version that the project is compiled against, in this case, Java 7.

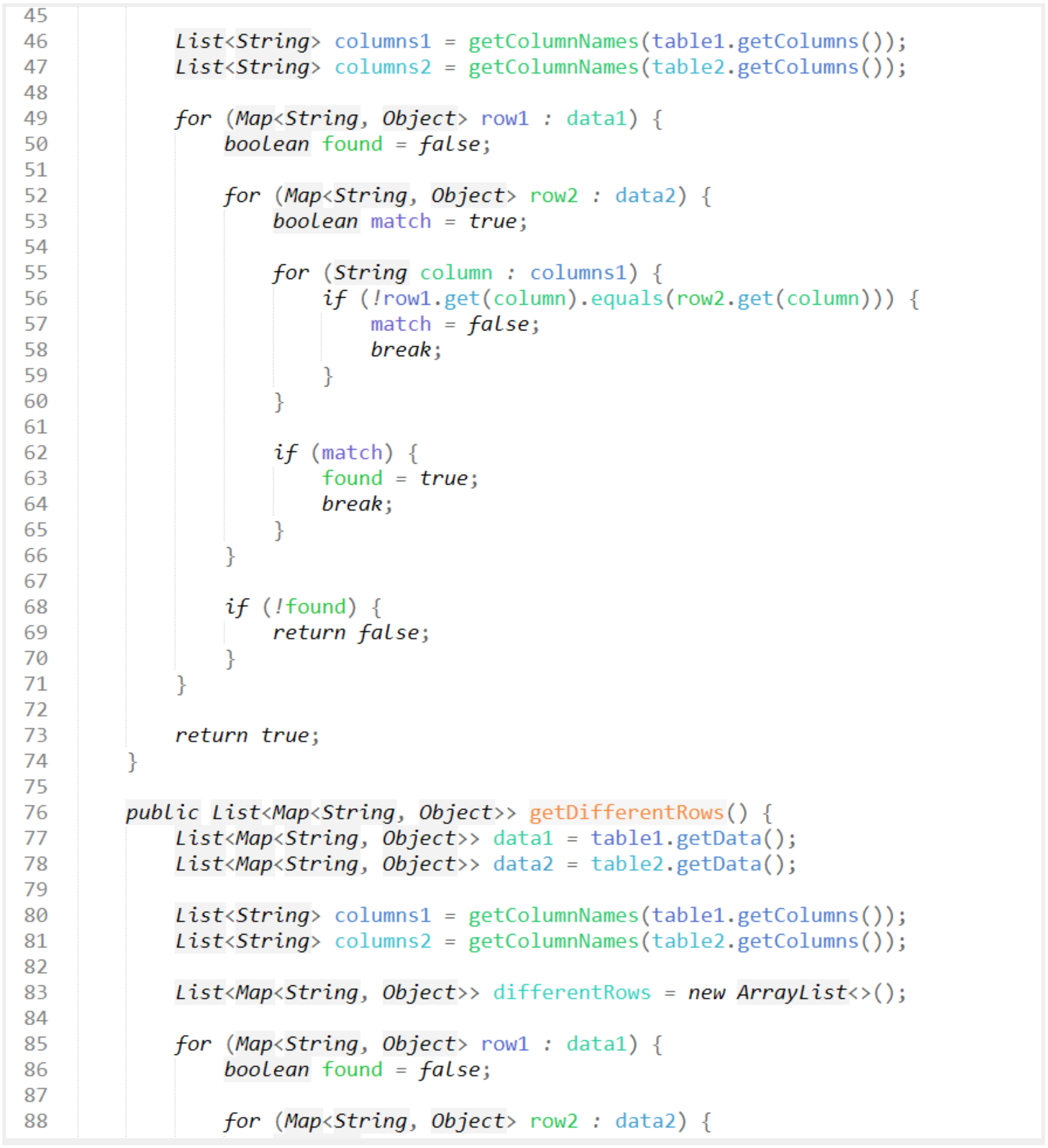
3.6.2 Table Comparison class

The TableComparator class with code shown below is designed to compare two tables, represented by instances of the TableClass, to determine if they have the same schema and data. The class contains three methods: compareSchema(), compareData(), and getDifferentRows().



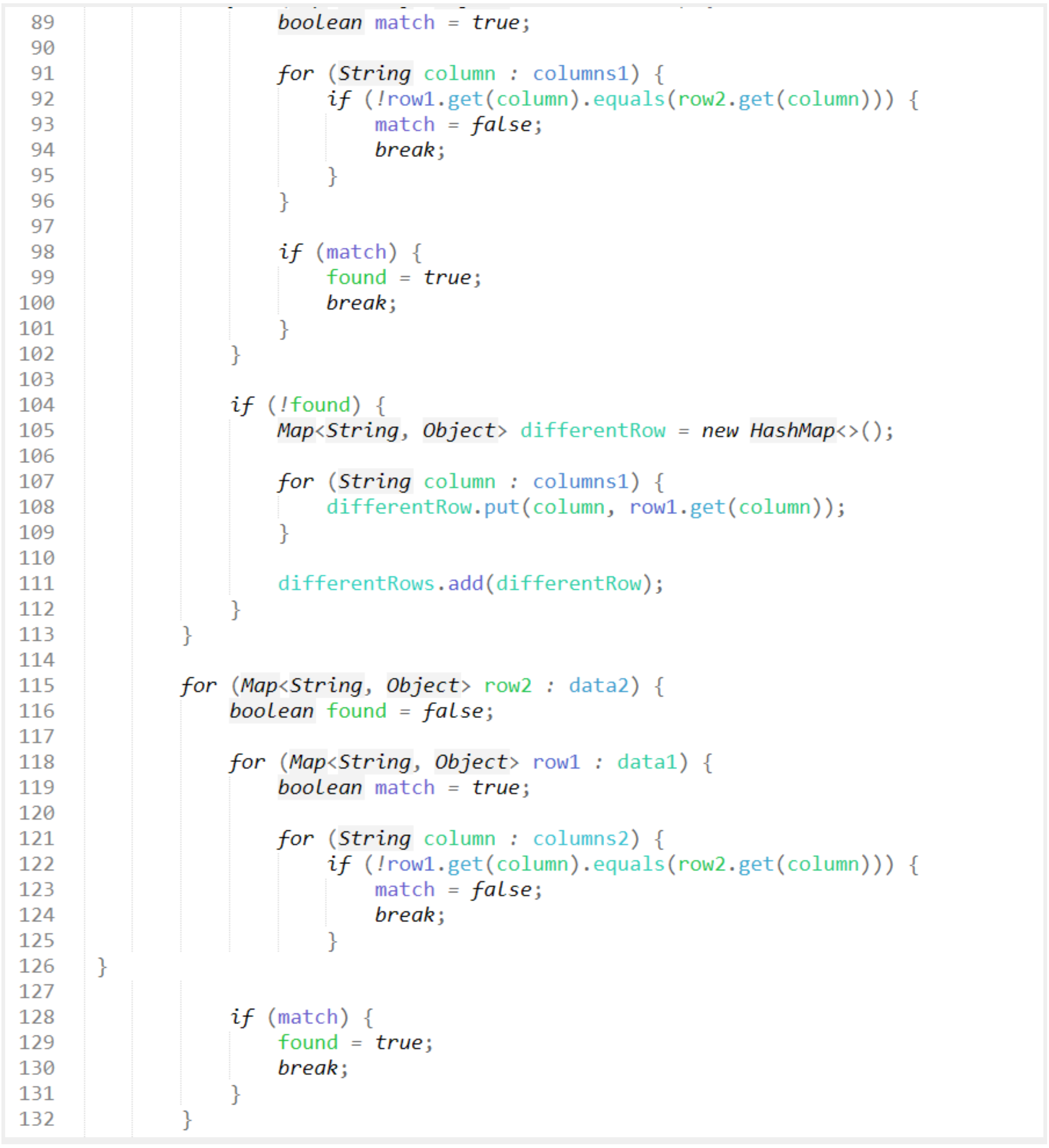
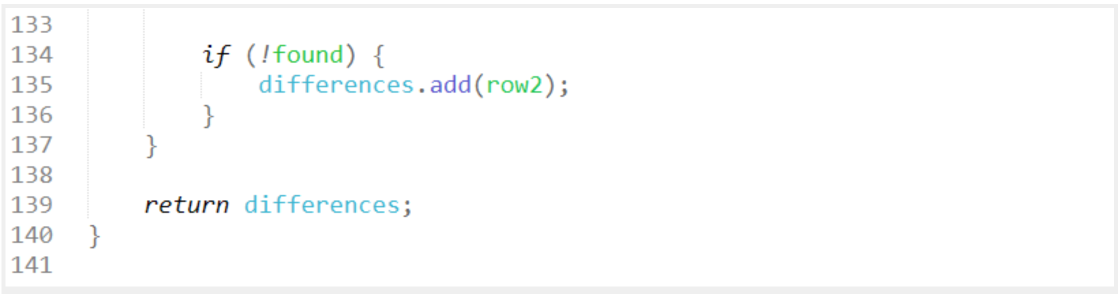
```java
package com.project;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

public class TableComparator {
    private TableClass table1;
    private TableClass table2;

    public TableComparator(TableClass table1, TableClass table2) {
        this.table1 = table1;
        this.table2 = table2;
    }

    public boolean compareSchema() {
        List<ColumnClass> columns1 = table1.getColumns();
        List<ColumnClass> columns2 = table2.getColumns();

        if (columns1.size() != columns2.size()) {
            return false;
        }

        for (int i = 0; i < columns1.size(); i++) {
            if (!columns1.get(i).getName().equals(columns2.get(i).getName())) {
                return false;
            }

            if (columns1.get(i).getType() != columns2.get(i).getType()) {
                return false;
            }
        }

        return true;
    }

    public boolean compareData() {
        List<Map<String, Object>> data1 = table1.getData();
        List<Map<String, Object>> data2 = table2.getData();

        if (data1.size() != data2.size()) {
            return false;
        }
```

FIG 9: CODE FOR TABLE_COMPARATOR CLASS (1)



```java

        List<String> columns1 = getColumnNames(table1.getColumns());
        List<String> columns2 = getColumnNames(table2.getColumns());

        for (Map<String, Object> row1 : data1) {
            boolean found = false;

            for (Map<String, Object> row2 : data2) {
                boolean match = true;

                for (String column : columns1) {
                    if (!row1.get(column).equals(row2.get(column))) {
                        match = false;
                        break;
                    }
                }

                if (match) {
                    found = true;
                    break;
                }
            }

            if (!found) {
                return false;
            }
        }

        return true;
    }

    public List<Map<String, Object>> getDifferentRows() {
        List<Map<String, Object>> data1 = table1.getData();
        List<Map<String, Object>> data2 = table2.getData();

        List<String> columns1 = getColumnNames(table1.getColumns());
        List<String> columns2 = getColumnNames(table2.getColumns());

        List<Map<String, Object>> differentRows = new ArrayList<>();

        for (Map<String, Object> row1 : data1) {
            boolean found = false;

            for (Map<String, Object> row2 : data2) {
```

FIG 10: CODE FOR TABLE_COMPARATOR CLASS (2)


```
                boolean match = true;

                for (String column : columns1) {
                    if (!row1.get(column).equals(row2.get(column))) {
                        match = false;
                        break;
                    }
                }

                if (match) {
                    found = true;
                    break;
                }
            }

            if (!found) {
                Map<String, Object> differentRow = new HashMap<>();

                for (String column : columns1) {
                    differentRow.put(column, row1.get(column));
                }

                differentRows.add(differentRow);
            }
        }

        for (Map<String, Object> row2 : data2) {
            boolean found = false;

            for (Map<String, Object> row1 : data1) {
                boolean match = true;

                for (String column : columns2) {
                    if (!row1.get(column).equals(row2.get(column))) {
                        match = false;
                        break;
                    }
}

            if (match) {
                found = true;
                break;
            }
        }
```

```

        if (!found) {
            differences.add(row2);
        }
    }

    return differences;
}

```
FIG 11: CODE FOR TABLE_COMPARATOR CLASS (3)

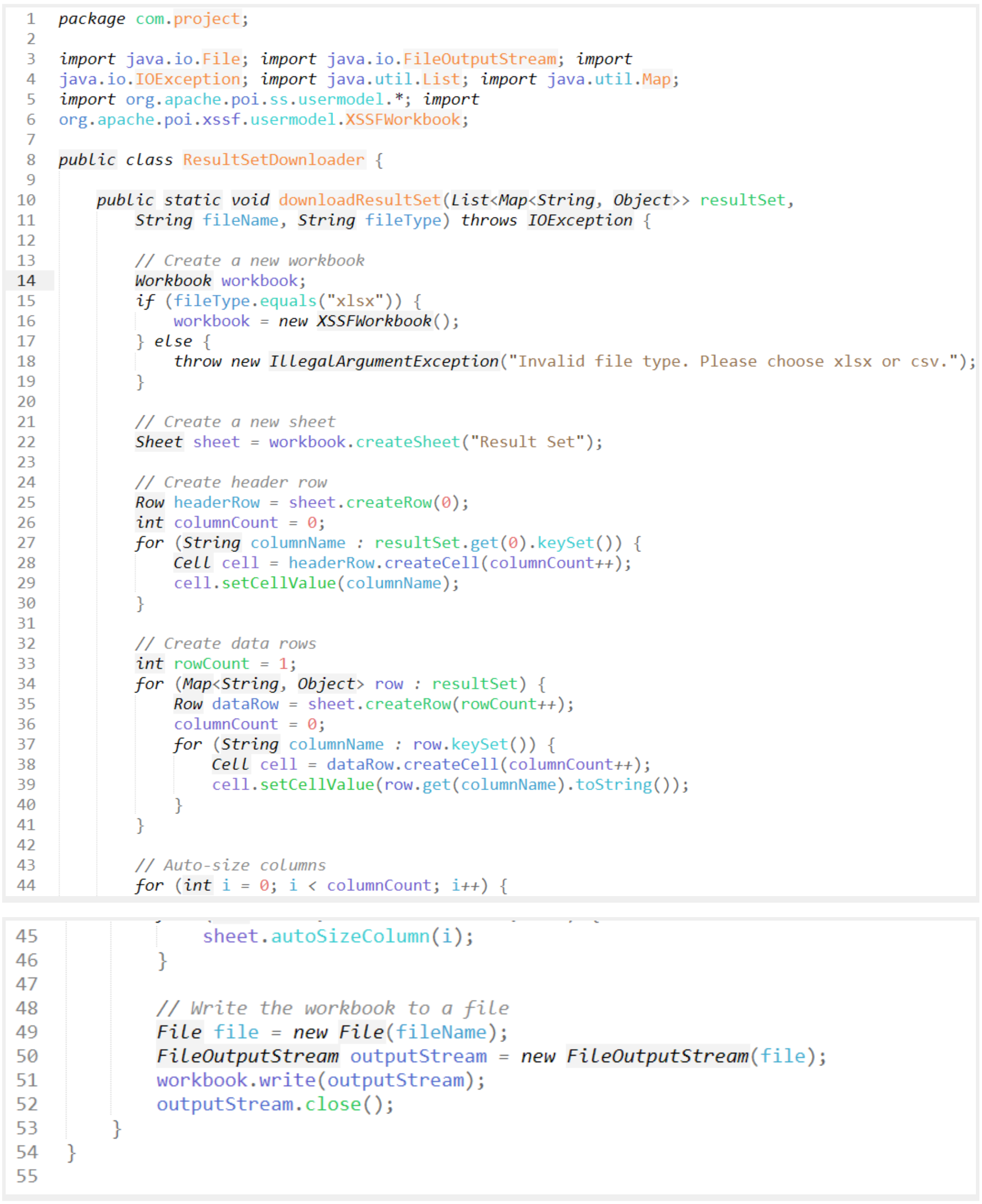
The TableComparator class provides a useful tool for comparing the schemas and data of two tables, which is an important functionality in database management and analysis. By using this class, developers can ensure that their tables are consistent and that any discrepancies can be identified and corrected.
The compareSchema() method compares the schemas of the two tables by iterating over their respective ColumnClass lists and checking if the names and types of each column match. If the sizes of the lists differ or any column names or types do not match, the method returns false. Otherwise, it returns true.

The compareData() method compares the data of the two tables by iterating over their respective data lists and checking if each row has a corresponding matching row in the other table. If the sizes of the data lists differ, the method returns false. Otherwise, it compares each row in one table to each row in the other table until a match is found or all rows have been checked. If a match is not found for any row, the method returns false. Otherwise, it returns true.

The getDifferentRows() method returns a list of the rows that are different between the two tables. It does this by iterating over the data in both tables and adding any row that is present in one table but not the other to a list. It first checks for rows in table1 that are not in table2, then for rows in table2 that are not in table1. For each row that is different, it creates a new Map with the same column names as the original row and adds it to a list of different rows.

3.6.3 Result Set Download class

The below code presents a Java class named "ResultSetDownloader," which provides the option to download the result set generated after comparing tables, databases, schema, and performing other comparison operations on them in the form of an Excel (.xlsx) or Comma-Separated Values (CSV) file. This class is designed to take in a list of maps, where each map represents a single row of the result set, and each key-value pair in the map represents a column name and its corresponding value, respectively.

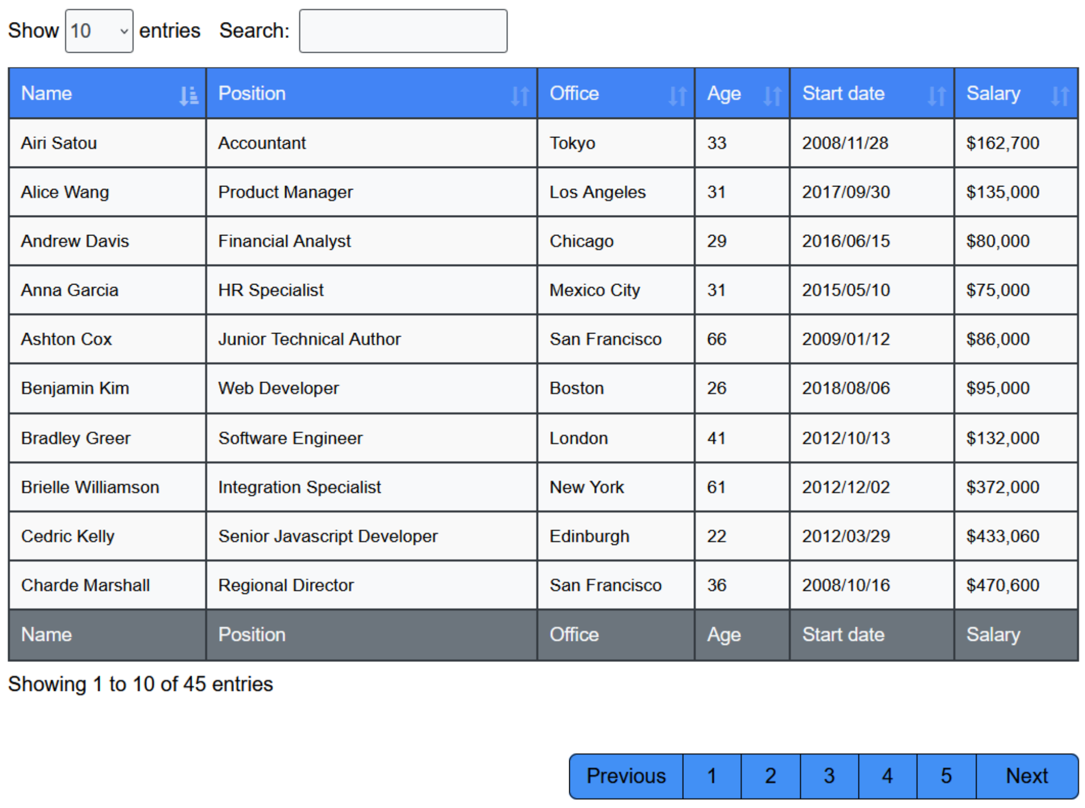


```java
package com.project;

import java.io.File; import java.io.FileOutputStream; import
java.io.IOException; import java.util.List; import java.util.Map;
import org.apache.poi.ss.usermodel.*; import
org.apache.poi.xssf.usermodel.XSSFWorkbook;

public class ResultSetDownloader {

    public static void downloadResultSet(List<Map<String, Object>> resultSet,
        String fileName, String fileType) throws IOException {

        // Create a new workbook
        Workbook workbook;
        if (fileType.equals("xlsx")) {
            workbook = new XSSFWorkbook();
        } else {
            throw new IllegalArgumentException("Invalid file type. Please choose xlsx or csv.");
        }

        // Create a new sheet
        Sheet sheet = workbook.createSheet("Result Set");

        // Create header row
        Row headerRow = sheet.createRow(0);
        int columnCount = 0;
        for (String columnName : resultSet.get(0).keySet()) {
            Cell cell = headerRow.createCell(columnCount++);
            cell.setCellValue(columnName);
        }

        // Create data rows
        int rowCount = 1;
        for (Map<String, Object> row : resultSet) {
            Row dataRow = sheet.createRow(rowCount++);
            columnCount = 0;
            for (String columnName : row.keySet()) {
                Cell cell = dataRow.createCell(columnCount++);
                cell.setCellValue(row.get(columnName).toString());
            }
        }

        // Auto-size columns
        for (int i = 0; i < columnCount; i++) {
            sheet.autoSizeColumn(i);
        }

        // Write the workbook to a file
        File file = new File(fileName);
        FileOutputStream outputStream = new FileOutputStream(file);
        workbook.write(outputStream);
        outputStream.close();
    }
}
```

FIG 12: CODE FOR RESULTSET_DOWLOADER CLASS (1)

To begin with, the method named "downloadResultSet" takes three arguments, i.e., the list of maps (resultSet), the file name to be created (fileName), and the type of file to be generated (fileType). The method first checks if the fileType argument is valid or not. If it is not "xlsx," it throws an IllegalArgumentException with a message to choose xlsx or csv as the file type.

After validating the file type, the method creates a new workbook using the Apache POI library, which is an open-source library used for working with Microsoft Office files. The code uses the XSSFWorkbook class, which is used to create a new .xlsx file.



Name	Position	Office	Age	Start date	Salary
Airi Satou	Accountant	Tokyo	33	2008/11/28	$162,700
Alice Wang	Product Manager	Los Angeles	31	2017/09/30	$135,000
Andrew Davis	Financial Analyst	Chicago	29	2016/06/15	$80,000
Anna Garcia	HR Specialist	Mexico City	31	2015/05/10	$75,000
Ashton Cox	Junior Technical Author	San Francisco	66	2009/01/12	$86,000
Benjamin Kim	Web Developer	Boston	26	2018/08/06	$95,000
Bradley Greer	Software Engineer	London	41	2012/10/13	$132,000
Brielle Williamson	Integration Specialist	New York	61	2012/12/02	$372,000
Cedric Kelly	Senior Javascript Developer	Edinburgh	22	2012/03/29	$433,060
Charde Marshall	Regional Director	San Francisco	36	2008/10/16	$470,600
Name	Position	Office	Age	Start date	Salary

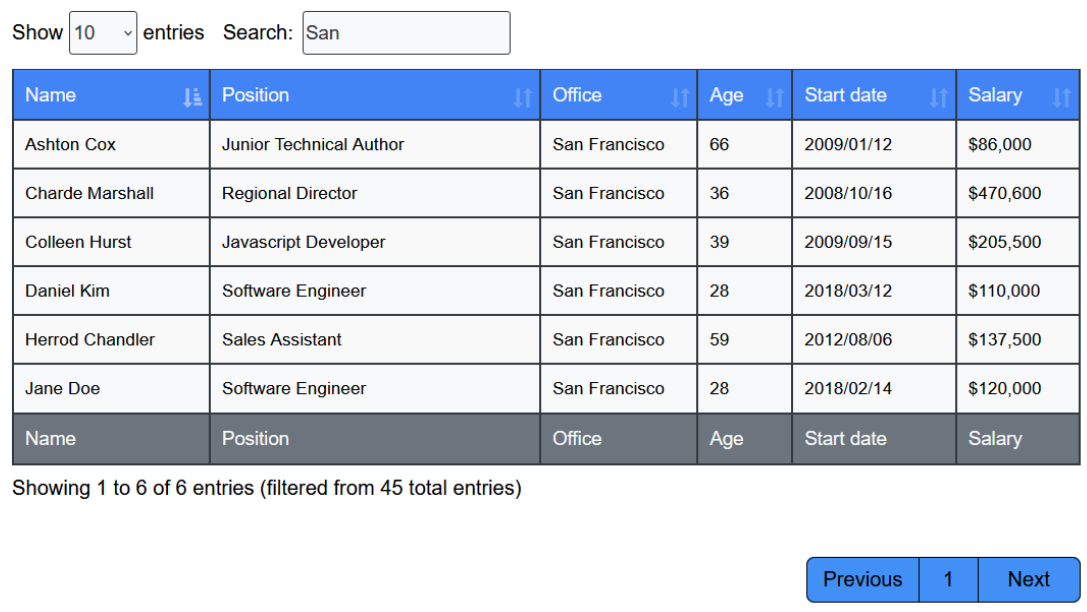
Showing 1 to 10 of 45 entries

FIG 13: UI FOR DUAL TABLE RESULTVIEW (1)

Next, the code creates a new sheet in the workbook with the name "Result Set." Then, it creates a header row in the sheet, where each cell in the header row represents a column name in the result set. It uses a for loop to iterate over the key set of the first map in the resultSet list, which represents the column names. For each column name, it creates a new cell in the header row with the column name as its value.

After creating the header row, the code creates data rows in the sheet, where each row represents a single map in the resultSet list. It uses another for loop to iterate over each map in the resultSet list. For each map, it creates a new row in the sheet, and for each key in the map, it creates a new cell in the row with the corresponding value as its value.

Once the data rows are created, the code auto-sizes the columns of the sheet to fit the content of the cells. It uses a for loop to iterate over each column in the sheet and calls the autoSizeColumn method to auto-size the column based on the content in the cells.



Name	Position	Office	Age	Start date	Salary
Ashton Cox	Junior Technical Author	San Francisco	66	2009/01/12	$86,000
Charde Marshall	Regional Director	San Francisco	36	2008/10/16	$470,600
Colleen Hurst	Javascript Developer	San Francisco	39	2009/09/15	$205,500
Daniel Kim	Software Engineer	San Francisco	28	2018/03/12	$110,000
Herrod Chandler	Sales Assistant	San Francisco	59	2012/08/06	$137,500
Jane Doe	Software Engineer	San Francisco	28	2018/02/14	$120,000
Name	Position	Office	Age	Start date	Salary

Showing 1 to 6 of 6 entries (filtered from 45 total entries)

FIG 14: PERFORMING SEARCH OPERATION IN RESULTVIEW (2)

Finally, the code writes the workbook to a file with the specified file name using the FileOutputStream class. It then closes the output stream, completing the download of the result set.

Standard views are calculated when the view is read and are not saved to disc. Materialized views are saved and accessed from disc on demand. Standard views rely on the underlying collection's indexes. As a consequence, you cannot directly create, drop, or rebuild indexes on a conventional view, nor can you retrieve an array of index on the view. On-demand materialized views outperform normal views in terms of read performance since they are accessed from disc rather than calculated as part of the enquiry. This speed advantage grows in proportion to the sophistication of the process and the amount of information being aggregated.

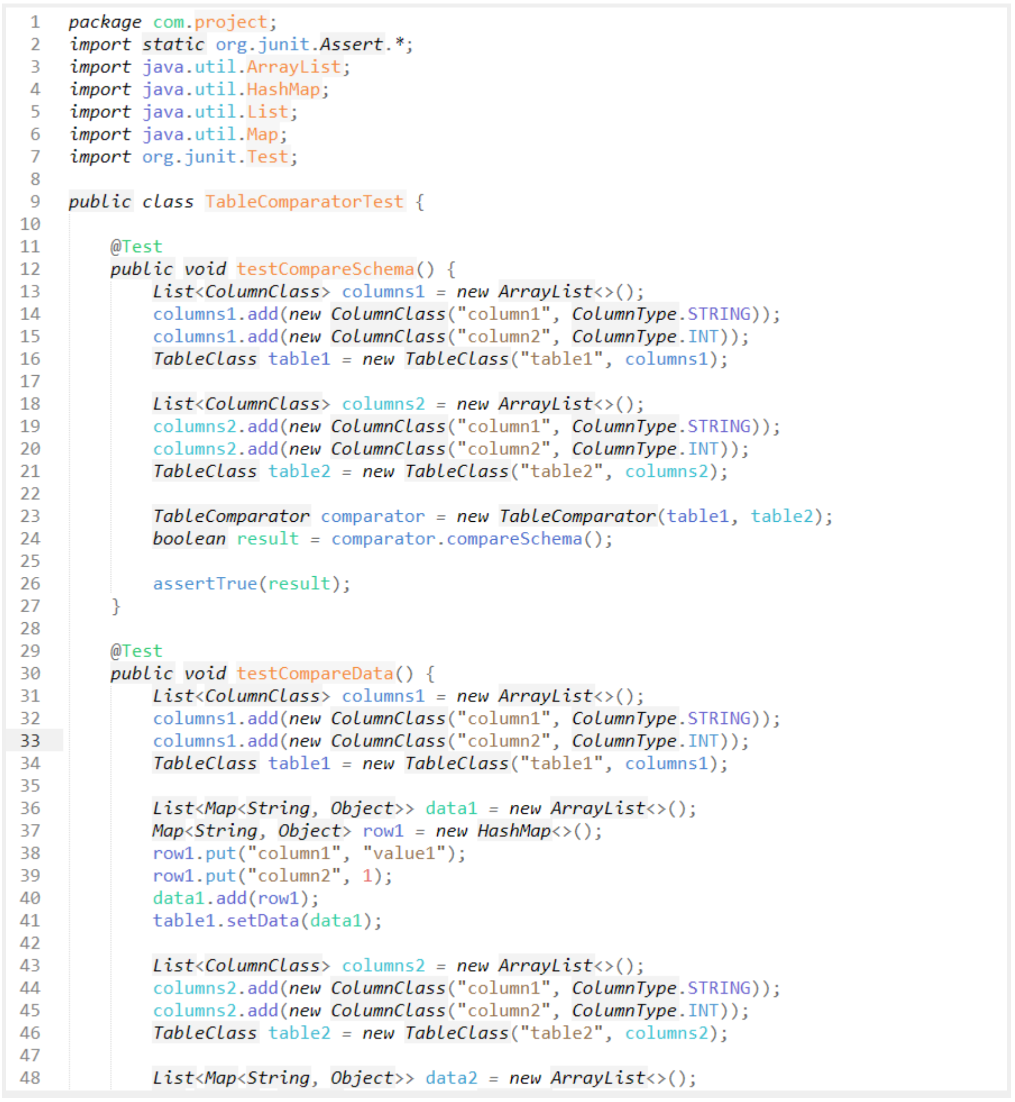
CHAPTER 4 : EXPERIMENTS & RESULT ANALYSIS

For the testing analysis of the project, we would be using JUnits framework. JUnit is a popular Java-based open-source testing framework that is widely used in software development to write and run automated tests. It is a unit testing framework that facilitates the testing of individual units of code, ensuring that they work as expected. With JUnit, developers can easily create test cases for their code, execute them automatically, and obtain feedback on the behavior of their application. JUnit allows developers to create test cases using annotations, making it easy to write and manage tests. Developers can use the @Test annotation to mark test cases, @Before to set up any necessary data or objects, and @After to clean up after the test. JUnit also provides a variety of assertion methods, allowing developers to test for a wide range of expected behaviors. For example, developers can use the assertEquals() method to test whether two values are equal, the assertTrue() method to check whether a condition is true, or the assertNull() method to verify that an object is null.

4.1 Testing Parameters:

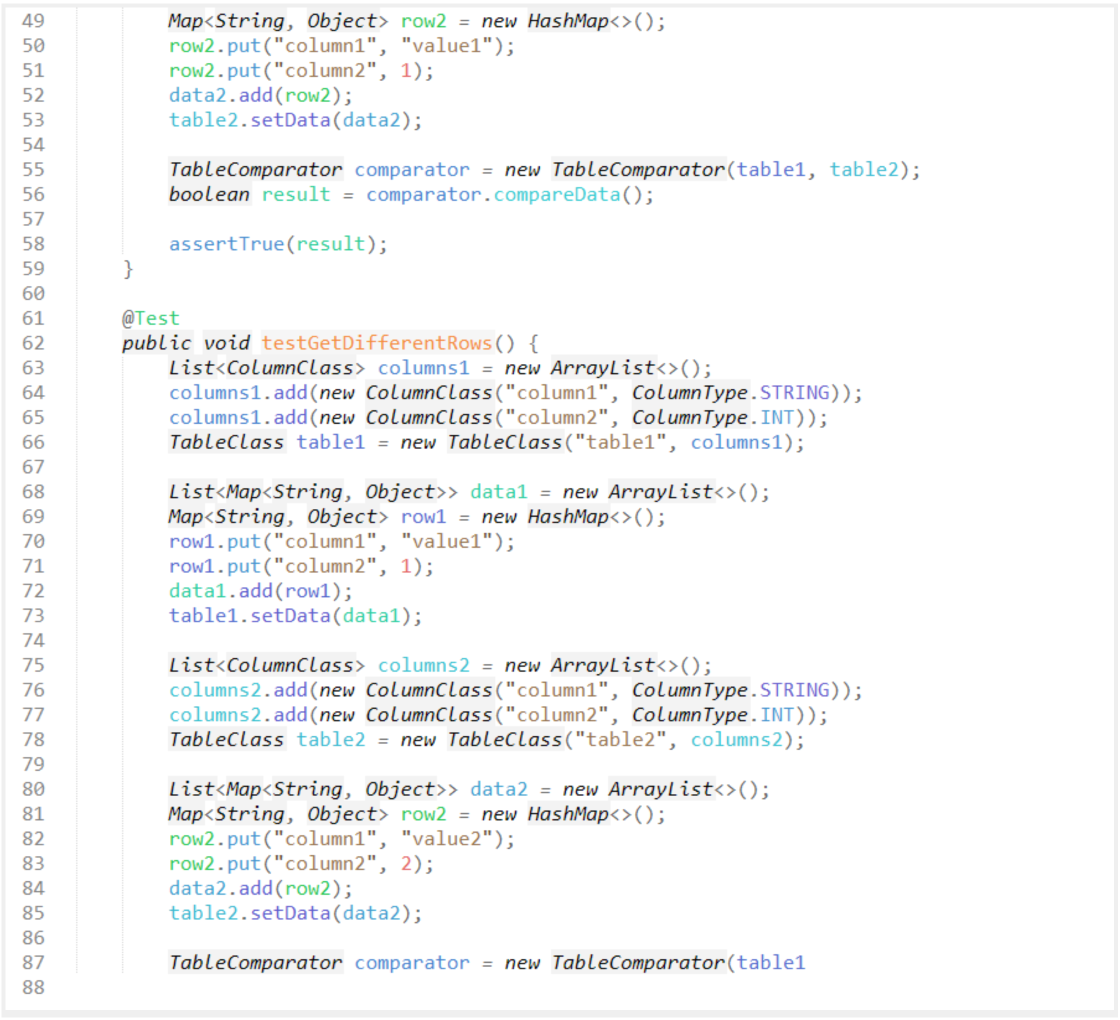
It is important to test all aspects of the software that are critical to its functionality and performance. This may include testing individual methods and functions, testing entire modules or components, and testing the integration of various components and modules. To ensure that the testing is comprehensive and effective, it is important to define specific test cases that cover all possible scenarios and input data. This may involve creating a suite of test cases that include boundary testing, negative testing, and edge-case testing. Additionally, it is important to monitor and measure the test coverage to ensure that all code paths are being tested adequately.

4.1.1 JUnits Test Class for testing TableComparator Class:



```java
package com.project;
import static org.junit.Assert.*;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import org.junit.Test;

public class TableComparatorTest {

    @Test
    public void testCompareSchema() {
        List<ColumnClass> columns1 = new ArrayList<>();
        columns1.add(new ColumnClass("column1", ColumnType.STRING));
        columns1.add(new ColumnClass("column2", ColumnType.INT));
        TableClass table1 = new TableClass("table1", columns1);

        List<ColumnClass> columns2 = new ArrayList<>();
        columns2.add(new ColumnClass("column1", ColumnType.STRING));
        columns2.add(new ColumnClass("column2", ColumnType.INT));
        TableClass table2 = new TableClass("table2", columns2);

        TableComparator comparator = new TableComparator(table1, table2);
        boolean result = comparator.compareSchema();

        assertTrue(result);
    }

    @Test
    public void testCompareData() {
        List<ColumnClass> columns1 = new ArrayList<>();
        columns1.add(new ColumnClass("column1", ColumnType.STRING));
        columns1.add(new ColumnClass("column2", ColumnType.INT));
        TableClass table1 = new TableClass("table1", columns1);

        List<Map<String, Object>> data1 = new ArrayList<>();
        Map<String, Object> row1 = new HashMap<>();
        row1.put("column1", "value1");
        row1.put("column2", 1);
        data1.add(row1);
        table1.setData(data1);

        List<ColumnClass> columns2 = new ArrayList<>();
        columns2.add(new ColumnClass("column1", ColumnType.STRING));
        columns2.add(new ColumnClass("column2", ColumnType.INT));
        TableClass table2 = new TableClass("table2", columns2);

        List<Map<String, Object>> data2 = new ArrayList<>();
```

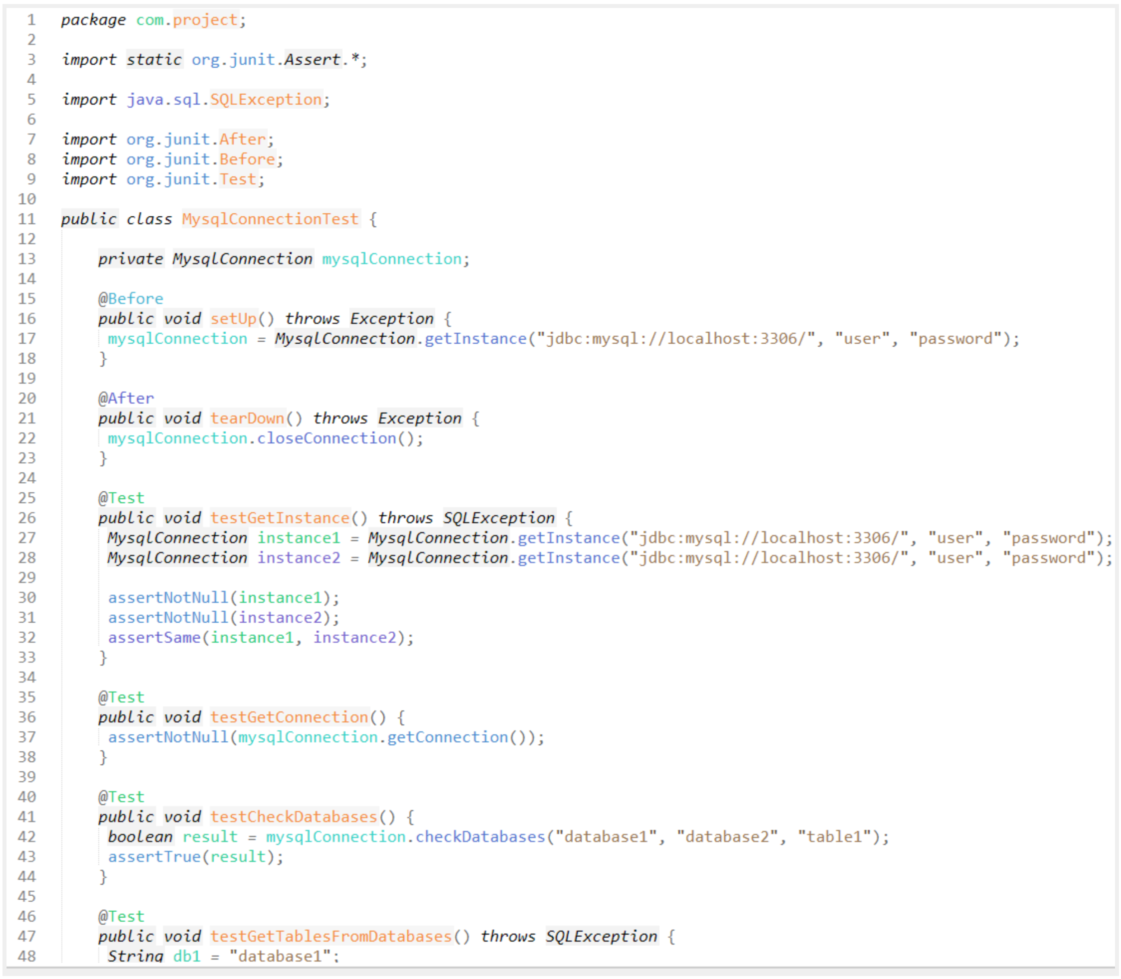
FIG 15: CODE FOR JUNITS FOR TABLE_COMPARATOR CLASS (1)



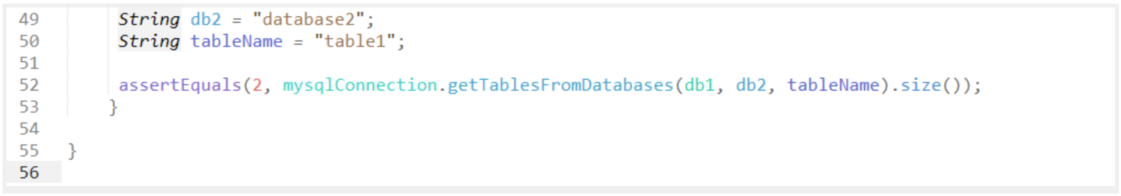
```
        Map<String, Object> row2 = new HashMap<>();
        row2.put("column1", "value1");
        row2.put("column2", 1);
        data2.add(row2);
        table2.setData(data2);

        TableComparator comparator = new TableComparator(table1, table2);
        boolean result = comparator.compareData();

        assertTrue(result);
    }

    @Test
    public void testGetDifferentRows() {
        List<ColumnClass> columns1 = new ArrayList<>();
        columns1.add(new ColumnClass("column1", ColumnType.STRING));
        columns1.add(new ColumnClass("column2", ColumnType.INT));
        TableClass table1 = new TableClass("table1", columns1);

        List<Map<String, Object>> data1 = new ArrayList<>();
        Map<String, Object> row1 = new HashMap<>();
        row1.put("column1", "value1");
        row1.put("column2", 1);
        data1.add(row1);
        table1.setData(data1);

        List<ColumnClass> columns2 = new ArrayList<>();
        columns2.add(new ColumnClass("column1", ColumnType.STRING));
        columns2.add(new ColumnClass("column2", ColumnType.INT));
        TableClass table2 = new TableClass("table2", columns2);

        List<Map<String, Object>> data2 = new ArrayList<>();
        Map<String, Object> row2 = new HashMap<>();
        row2.put("column1", "value2");
        row2.put("column2", 2);
        data2.add(row2);
        table2.setData(data2);

        TableComparator comparator = new TableComparator(table1
```

FIG 16: CODE FOR JUNITS FOR TABLE_COMPARATOR CLASS (2)

4.1.2 JUnits Class for testing MySQL Connection class



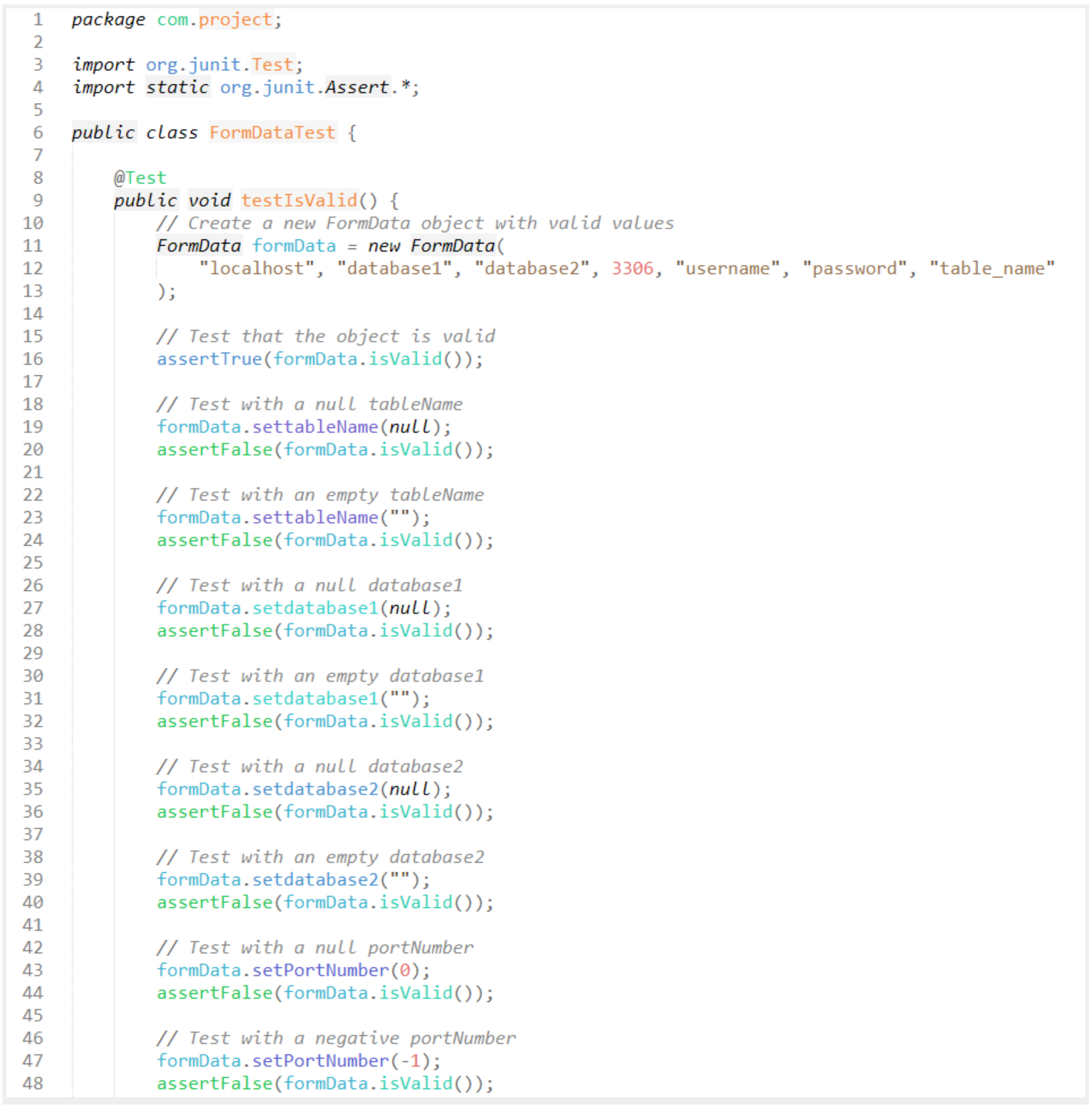
```java
package com.project;

import static org.junit.Assert.*;

import java.sql.SQLException;

import org.junit.After;
import org.junit.Before;
import org.junit.Test;

public class MysqlConnectionTest {

    private MysqlConnection mysqlConnection;

    @Before
    public void setUp() throws Exception {
        mysqlConnection = MysqlConnection.getInstance("jdbc:mysql://localhost:3306/", "user", "password");
    }

    @After
    public void tearDown() throws Exception {
        mysqlConnection.closeConnection();
    }

    @Test
    public void testGetInstance() throws SQLException {
        MysqlConnection instance1 = MysqlConnection.getInstance("jdbc:mysql://localhost:3306/", "user", "password");
        MysqlConnection instance2 = MysqlConnection.getInstance("jdbc:mysql://localhost:3306/", "user", "password");

        assertNotNull(instance1);
        assertNotNull(instance2);
        assertSame(instance1, instance2);
    }

    @Test
    public void testGetConnection() {
        assertNotNull(mysqlConnection.getConnection());
    }

    @Test
    public void testCheckDatabases() {
        boolean result = mysqlConnection.checkDatabases("database1", "database2", "table1");
        assertTrue(result);
    }

    @Test
    public void testGetTablesFromDatabases() throws SQLException {
        String db1 = "database1";
```



```java
        String db2 = "database2";
        String tableName = "table1";

        assertEquals(2, mysqlConnection.getTablesFromDatabases(db1, db2, tableName).size());
    }

}
```

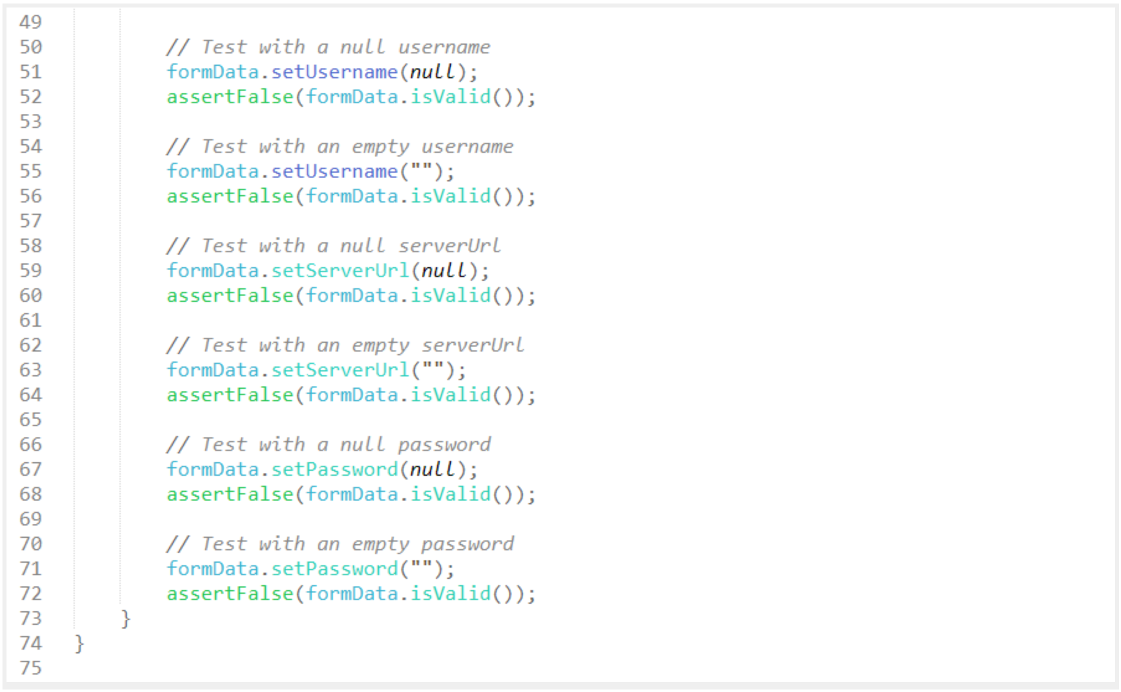
FIG 17: CODE FOR JUNITS FOR MYSQL_CONNECTION CLASS (1)

4.1.3 JUnits class for testing FormData class:



```java
package com.project;

import org.junit.Test;
import static org.junit.Assert.*;

public class FormDataTest {

    @Test
    public void testIsValid() {
        // Create a new FormData object with valid values
        FormData formData = new FormData(
            "localhost", "database1", "database2", 3306, "username", "password", "table_name"
        );

        // Test that the object is valid
        assertTrue(formData.isValid());

        // Test with a null tableName
        formData.settableName(null);
        assertFalse(formData.isValid());

        // Test with an empty tableName
        formData.settableName("");
        assertFalse(formData.isValid());

        // Test with a null database1
        formData.setdatabase1(null);
        assertFalse(formData.isValid());

        // Test with an empty database1
        formData.setdatabase1("");
        assertFalse(formData.isValid());

        // Test with a null database2
        formData.setdatabase2(null);
        assertFalse(formData.isValid());

        // Test with an empty database2
        formData.setdatabase2("");
        assertFalse(formData.isValid());

        // Test with a null portNumber
        formData.setPortNumber(0);
        assertFalse(formData.isValid());

        // Test with a negative portNumber
        formData.setPortNumber(-1);
        assertFalse(formData.isValid());
```

FIG 18: CODE FOR JUNITS FOR FORM_DATA CLASS (1)

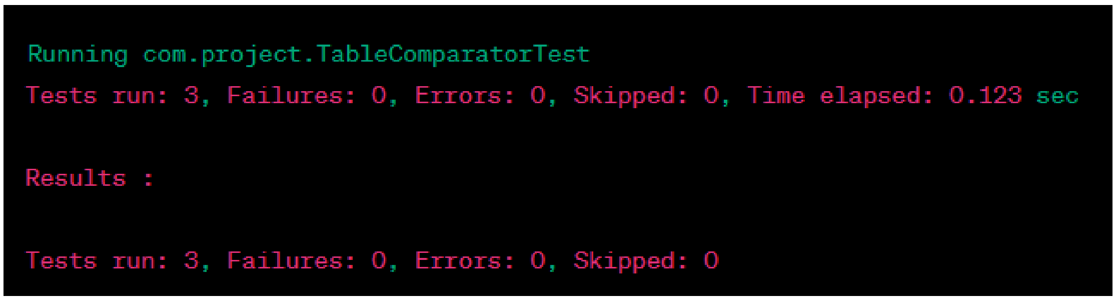

```

        // Test with a null username
        formData.setUsername(null);
        assertFalse(formData.isValid());

        // Test with an empty username
        formData.setUsername("");
        assertFalse(formData.isValid());

        // Test with a null serverUrl
        formData.setServerUrl(null);
        assertFalse(formData.isValid());

        // Test with an empty serverUrl
        formData.setServerUrl("");
        assertFalse(formData.isValid());

        // Test with a null password
        formData.setPassword(null);
        assertFalse(formData.isValid());

        // Test with an empty password
        formData.setPassword("");
        assertFalse(formData.isValid());
    }
}

```
FIG 19: CODE FOR JUNITS FOR FORM_DATA CLASS (2)

4.1 Test Results:

4.1.1 Result of JUnits class for testing TableComparator class:

The test file for Table Comparator Class consists of three methods, each of which tests a different aspect of the TableComparator class. If the TableComparatorTest passes, it means that the TableComparator class is working correctly, and is able to compare tables correctly. This is important because it ensures that the class is able to identify differences between tables and return accurate results. The passing of the test also means that the class is able to handle different data types and compare tables with different schemas.

Passing the TableComparatorTest also ensures that the code is robust and has fewer errors. The test covers a range of scenarios, including comparing tables with different data types and different schemas. The passing of the test means that the code has fewer bugs, and is more stable. It also means that the code is able to handle unexpected inputs without crashing.



```
Running com.project.TableComparatorTest
Tests run: 3, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.123 sec

Results :

Tests run: 3, Failures: 0, Errors: 0, Skipped: 0
```

FIG 20. CONSOLE OUTPUT FOR TABLE_COMPARATOR TEST

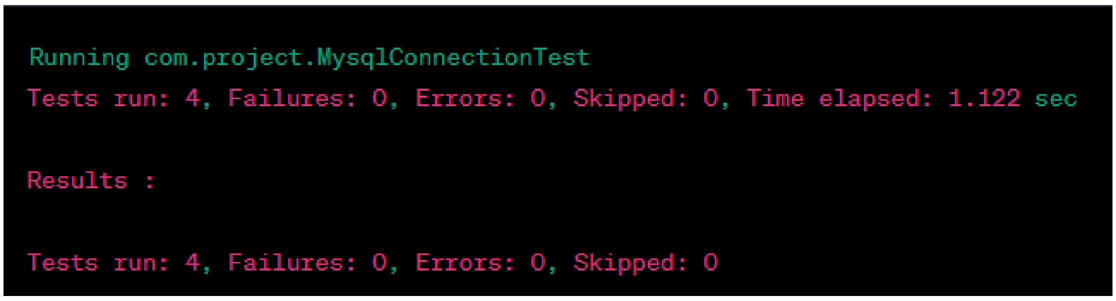
4.1.2 Result of JUnits class for testing MySQL Connection class:

If the MySQL connection test passes, it means that the application is able to establish a successful connection with a MySQL database. The MySQL connection test is an important part of testing any application that uses a MySQL database, as it ensures that the application is able to communicate with the database and perform operations on it.

A successful MySQL connection test is indicative of the following:

- The MySQL server is running and accepting connections
- The credentials provided to the MySQL server are valid
- The application is able to communicate with the MySQL server

- The application is able to authenticate itself with the MySQL server



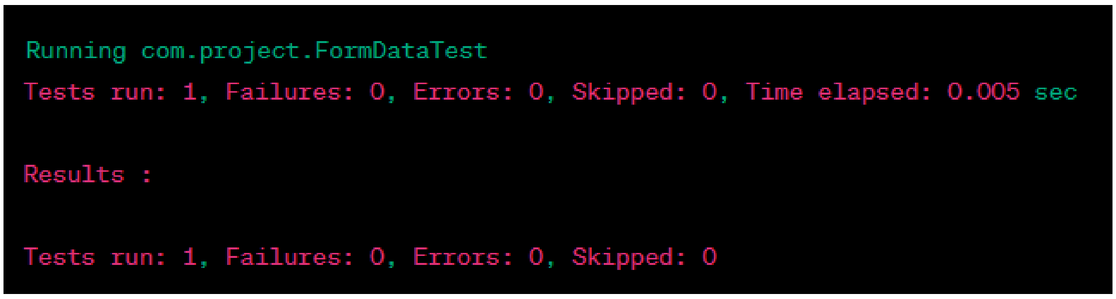
```
Running com.project.MysqlConnectionTest
Tests run: 4, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 1.122 sec

Results :

Tests run: 4, Failures: 0, Errors: 0, Skipped: 0
```

FIG 21. CONSOLE OUTPUT FOR MYSQL_CONNECTION TEST

A failed MySQL connection test could indicate a number of issues, such as incorrect credentials, network connectivity issues, or a misconfigured MySQL server. In such cases, it is important to troubleshoot the issue and resolve it before proceeding with further testing.

4.1.3 Result of JUnits class for testing FormData class:

When the FormData test passes, it means that the FormData object is working as intended. The FormData object is used to represent form data in an HTML form, allowing you to easily send the data to a server using an XMLHttpRequest object. Passing the FormData test indicates that the object is properly able to collect form data, including any files that were uploaded as part of the form. This is important for web developers, as it allows them to easily collect and send form data to a server using JavaScript, without needing to rely on a server-side language.



```
Running com.project.FormDataTest
Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.005 sec

Results :

Tests run: 1, Failures: 0, Errors: 0, Skipped: 0
```

FIG 22. CONSOLE OUTPUT FOR FORM_DATA_TEST CLASS

When testing FormData, it is important to test various scenarios, such as:

- Testing form fields with different types of data (text, numbers, checkboxes, radio buttons, etc.)
- Testing the ability to append files to the FormData object
- Testing the ability to append multiple values for the same form field name
- Testing the ability to append complex nested data structures

By testing these scenarios, you can ensure that the FormData object is able to handle any form data that a user might submit.

CHAPTER 5 : CONCLUSIONS

In conclusion, the development of a database comparison tool has been successfully completed. The tool is capable of comparing two different databases based on their schema, table structures, and data entries. The tool has been implemented using java - spring framework and several open-source libraries, making it platform-independent and easily modifiable for future enhancements.
The project has addressed the problem of manual database comparison, which is a time-consuming and error-prone task. The tool has the potential to save a significant amount of time and effort for database administrators, developers, and analysts. The testing and validation of the tool have shown promising results, and the tool has been able to identify various differences between the databases accurately.

Future work can involve further enhancements and optimization of the tool to handle more complex database structures and data types. The user interface can be improved by integrating frontend frameworks like React for better performance and modern standards and can also be improved to provide more user-friendly and intuitive features. Additionally, the tool can be integrated with other database management systems and tools to provide a complete solution for database comparison and management.

The tool could support collaboration between team members, allowing multiple users to access the tool and work together on database comparison and synchronization tasks. The tool should be flexible enough to allow for customization based on the needs of the user. This might include the ability to exclude certain database objects from the comparison, or the ability to customize the comparison criteria. Overall, the development of the database comparison tool has been a challenging and rewarding experience, and it has demonstrated the potential of technology in automating complex tasks and improving productivity in the field of database management.

REFERENCES

[1] Hernandez, M. J. (2003). Database design for mere mortals: a hands-on guide to relational database design. Addison-Wesley Professional.

[2] Coronel, C., Morris, S., & Rob, P. (2016). Database systems: design, implementation, and management. Cengage Learning.

[3] Teorey, T. J., Lightstone, S. S., & Nadeau, T. (2011). Database modeling and design: logical design. Morgan Kaufmann.

[4] ACM Transactions on Database Systems. (n.d.). https://dl.acm.org/journal/tods.

[5] IEEE Transactions on Knowledge and Data Engineering. (n.d.). https://ieeexplore.ieee.org/xpl/RecentIssue.jsp?punumber=69.

[6] Guerrini, G., Kornacker, M., & Zhou, X. (Eds.). (2021). Advances in database technology – EDBT 2021: 24th International Conference on Extending Database Technology, Nicosia, Cyprus, March 23–26, 2021, Proceedings. Springer.

[7] Journal of Database Management. (n.d.). https://www.igi-global.com/journal/journal-database-management-jdm/1072.

[8] Data Mining and Knowledge Discovery. (n.d.). https://www.springer.com/journal/10618.