CSCI 5408

ASSIGNMENT 2

Aharnish Solanki

B00933563

Summary

The research paper titled "Comparative Analysis of Data Fragmentation in Distributed Database" presents a comprehensive analysis into various data fragmentation techniques within distributed database systems. The paper gives a detailed introduction to distributed databases, emphasizing their increasing significance in a wide range of networked environments. It focuses on the increasing potential of developing distributed database systems as communication technology, software, and hardware advance. The paper mainly focuses on three types of data fragmentation: horizontal fragmentation (HF), vertical fragmentation (VF), and mixed fragmentation (MF).

Horizontal fragmentation is the process of dividing a relation or class into disjoint tuples or instances, each of which is stored at a different node in the distributed system. The primary objective of horizontal fragmentation is to ensure that each site possesses the necessary information required for efficient query processing. Horizontal fragmentation improves query performance and reduces disc accesses by reducing the retrieval of irrelevant data by applications.

On the other hand, vertical fragmentation divides a class or relation into separate sets of columns or attributes, with each set typically sharing at least one common attribute, often the primary key. This method facilitates different sites to handle different functions related to the fragmented entity. Vertical fragmentation provides several benefits, including increased efficiency through the use of localized query optimization techniques and reduced data access by removing irrelevant data from each site. However, the paper acknowledges the potential difficulties that can arise when data from different fragments needs to be accessed, as this may introduce additional communication costs within the distributed system.

The third type of fragmentation i.e., the mixed fragmentation, combines elements of both horizontal and vertical strategies. It involves dividing a table into horizontal subsets and then applying vertical fragmentation within each horizontal fragment. Mixed fragmentation allows for greater flexibility in accommodating specific requirements and optimizing query performance. However, the paper acknowledges that mixed fragmentation is frequently regarded as the most complex approach, requiring careful management and coordination to ensure successful implementation.

The paper thoroughly examines the benefits and drawbacks of each fragmentation type. Horizontal fragmentation, for instance, improves query performance by storing relevant data at each site. However, in cases involving recursive fragmentation, it might need costly reconstruction processes. Vertical fragmentation improves efficiency through localized query optimization and reduced data access, but it can introduce communication costs when data from disparate fragments needs to be accessed. Mixed fragmentation combines the benefits of both horizontal and vertical approaches but its implementation complexity should be carefully managed.

The paper emphasizes on the importance of considering multiple factors when making decisions about fragmentation. It focuses on the importance of both quantitative and qualitative data in determining the best fragmentation strategy. Quantitative factors, such as relation cardinality, query frequency, and selectivity play a crucial role in evaluating the performance consequences of fragmentation choices. Qualitative factors, such as query predicates and data access types, provide insight into the specific requirements and characteristics of the distributed database environment.

Furthermore, the paper focuses on the importance of completeness and reconstruction in fragmentation. Completeness ensures that each data item in the original relation can be found in at least one fragment, guaranteeing the integrity and coherence of the data. And, Reconstruction refers to the availability of relational operators capable of reconstructing the original relation from its fragments, ensuring flexibility in retrieving the complete dataset when necessary.

The comprehensive analysis of data fragmentation strategies in distributed database systems offers valuable insights to researchers and practitioners in the field. It serves as an invaluable resource for new researchers seeking to explore the potential of distributed databases and make informed decisions regarding data fragmentation.

In conclusion, the research paper significantly contributes to the understanding of data fragmentation in distributed database systems. The paper provides researchers with the advantages, disadvantages, and comparisons of different fragmentation types necessary for creating effective distributed databases. The insights provided paves way for further advancements and improvements in distributed database technology, ensuring maximum performance and resource utilization.

References:

[1] A. H. Al-Sanhani, A. Hamdan, A. B. Al-Thaher and A. Al-Dahoud, "A comparative analysis of data fragmentation in distributed database," 2017 8th International Conference on Information Technology (ICIT), Amman, Jordan, 2017, pp. 724-729, doi: 10.1109/ICITECH.2017.8079934.

CSCI 5408

Assignment 2:

Problem 2- Prototype of a light-weight DBMS using Java programming language.

Aharnish Solanki

B00933563

Problem 2- Prototype of a light-weight DBMS using Java programming language

Summary:

This project adresses to develop a light-weight database management system (DBMS) prototype using the Java programming language. The prototype should not rely on any third-party libraries or frameworks. The goal is to create a tool similar to MySQL but implemented from scratch. The development of the application should be done using a standard Java IDE, and any version of the Java Development Kit (JDK) is acceptable.

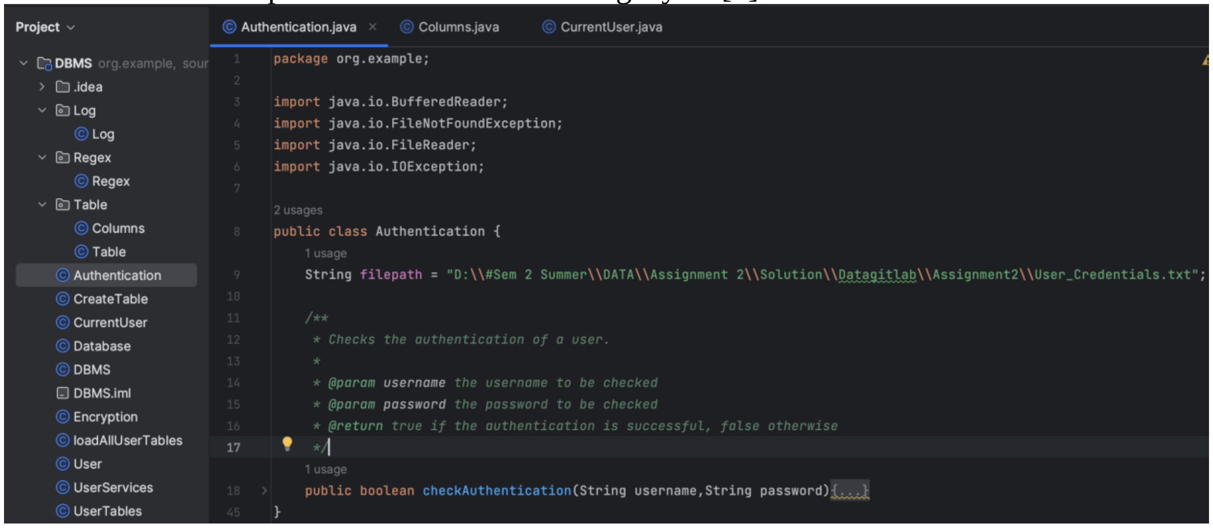
When writing the Java code for the application, it is important to follow the JavaDocs specification for commenting styles. The application is using the Java Doc standards for better readability and maintainability of the codebase. As part of the application program development and execution, it is fulfilling the design principles. The SOLID design principles are used as a guide for designing the application such as created separate classes for each database operations.

The application has a console-based interface, and a graphical user interface (GUI) is not implemented. Users is able to input SQL queries after successfully logging in. The application provides functionality for creating only one database. One of the required functionalities of the application is implementing two-factor authentication for user authentication. This authentication module uses the user's ID, password, and a question/answer combination for verification. It is suggested to design a specific class to handle this authentication process. The purpose of this authentication is to support multiple users accessing the application.

The second required functionality is the design of persistent storage. Once an input query is processed, the data, user information, and logs are stored in a custom-designed file format. Essential requirement is the implementation of various SQL queries, including Data Definition Language (DDL) and Data Manipulation Language (DML) operations. This includes creating tables (CREATE), retrieving data (SELECT), inserting data (INSERT), updating data (UPDATE), and deleting data (DELETE) from any number of tables.

Tasks Done:

- Standard Java IDE used: eclipse [1]
- Java version: 17
- Used the JavaDocs specification for commenting styles [2]:

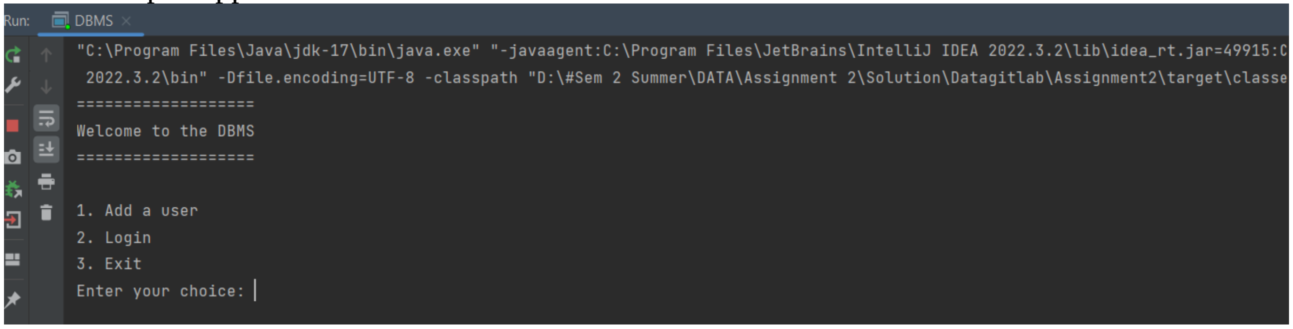


```
package org.example;

import java.io.BufferedReader;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.IOException;

2 usages
public class Authentication {
    1 usage
    String filepath = "D:\\#Sem 2 Summer\\DATA\\Assignment 2\\Solution\\Datagitlab\\Assignment2\\User_Credentials.txt";

    /**
     * Checks the authentication of a user.
     *
     * @param username the username to be checked
     * @param password the password to be checked
     * @return true if the authentication is successful, false otherwise
     */
    1 usage
    public boolean checkAuthentication(String username,String password){...}
}
```

Figure 1: Following JavaDocs specifications

- Using SOLID Principles:
    - Single Responsibility Principle (SRP)
        - The Authentication class has the responsibility of checking user authentication.
        - The CreateTable class has the responsibility of creating a table.
        - The CurrentUser class is responsible for managing the current user.
        - The UserTables class handles the creation of user-specific tables.
        - The Columns class represents a column in a table.

- The developed application is console based :



```
"C:\Program Files\Java\jdk-17\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2022.3.2\lib\idea_rt.jar=49915:C
 2022.3.2\bin" -Dfile.encoding=UTF-8 -classpath "D:\#Sem 2 Summer\DATA\Assignment 2\Solution\Datagitlab\Assignment2\target\classe
===================
Welcome to the DBMS
===================

1. Add a user
2. Login
3. Exit
Enter your choice: 
```

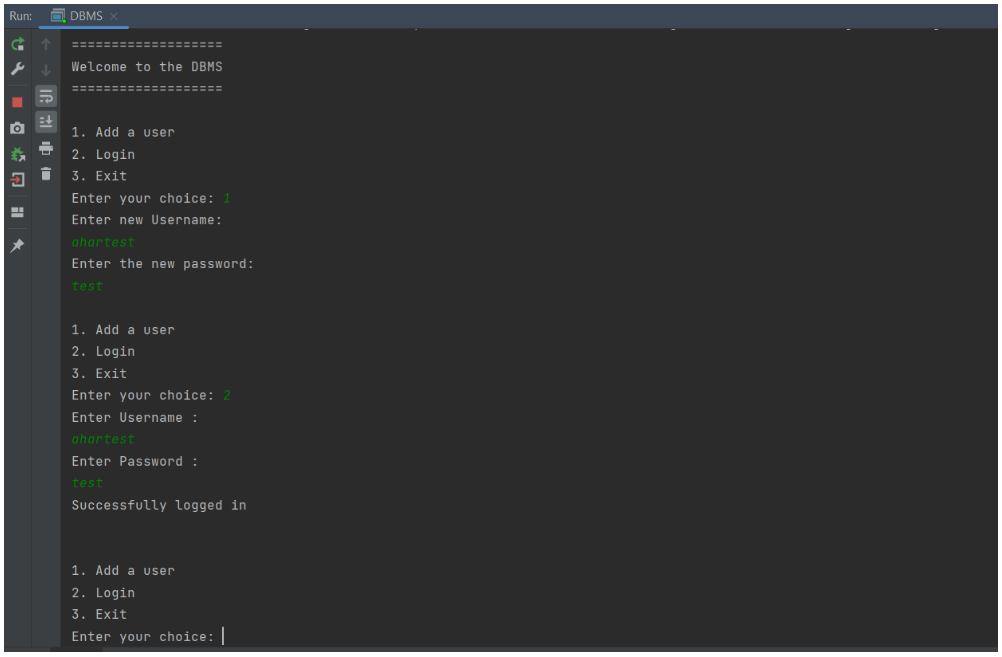
Figure 2: User interaction on Console.

- Two factor User Authentication:

It is a simplified implementation of a database management system with user authentication. It consists of three main classes: Encryption, Authentication, and DBMS.

The **Encryption** class handles password encryption using the MD5 algorithm. It provides a method called encryptPassword that takes an unhashed password as input, encrypts it using MD5 [3], and returns the hashed password as a string.

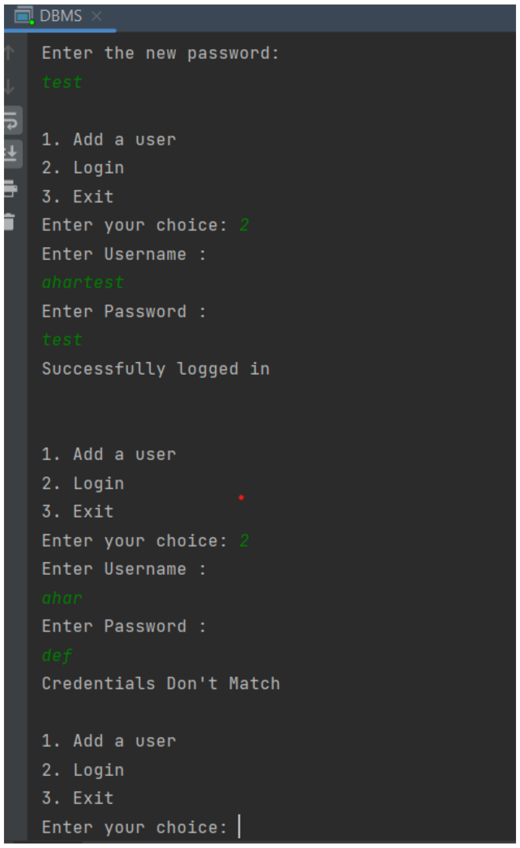
The **Authentication** class is responsible for checking the authentication of users. It reads user credentials from a file named "User_Credentials.txt" and compares them with the provided username and password. If a match is found, it returns true indicating successful authentication; otherwise, it returns false. The file format is "Username: <username> ~ Password: <password>;", where multiple user credentials are stored on separate lines.



```
==================
Welcome to the DBMS
==================

1. Add a user
2. Login
3. Exit
Enter your choice: 1
Enter new Username:
ahartest
Enter the new password:
test

1. Add a user
2. Login
3. Exit
Enter your choice: 2
Enter Username :
ahartest
Enter Password :
test
Successfully logged in


1. Add a user
2. Login
3. Exit
Enter your choice: 
```

Figure 3: User Registration

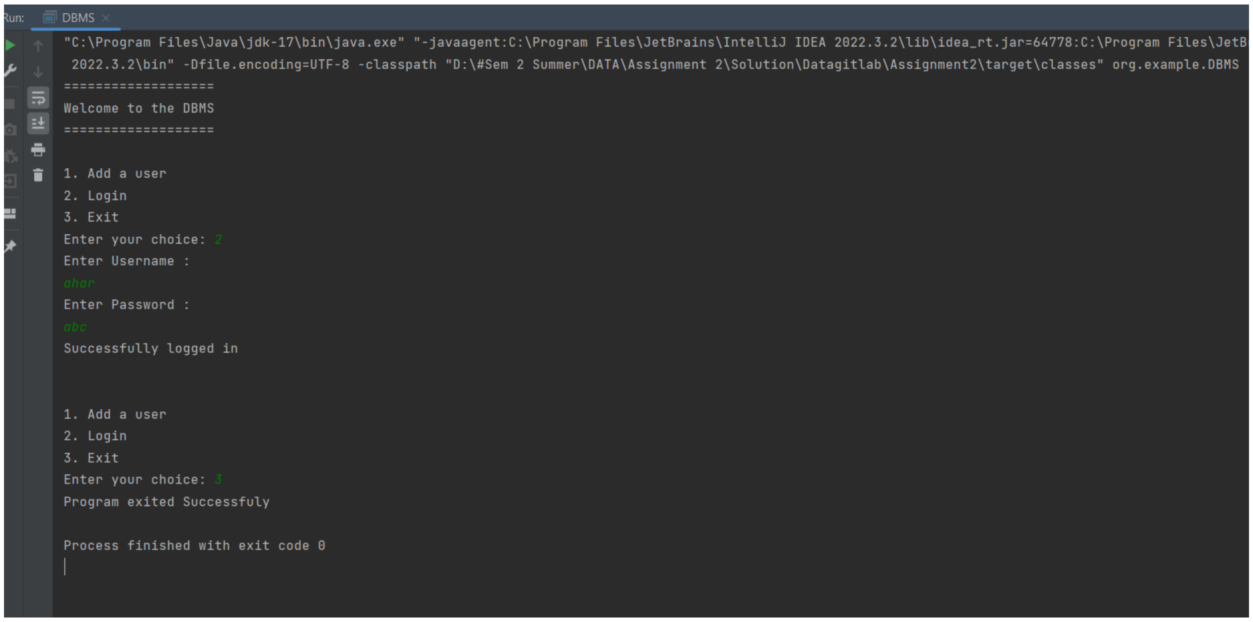
The DBMS class is the main entry point of the program. It creates instances of the Authentication, Encryption, and Log classes. It checks if the "User_Credentials.txt" file exists. If the file exists, it reads its contents. If the file is empty, it prompts the user to create a new user and adds the user's credentials to the file. If the file is not empty, it presents a menu with options to add a user, login, or exit.



```
Enter the new password:
test

1. Add a user
2. Login
3. Exit
Enter your choice: 2
Enter Username :
ahartest
Enter Password :
test
Successfully logged in


1. Add a user
2. Login
3. Exit
Enter your choice: 2
Enter Username :
ahar
Enter Password :
def
Credentials Don't Match

1. Add a user
2. Login
3. Exit
Enter your choice:
```

Figure 4: User Login mismatch

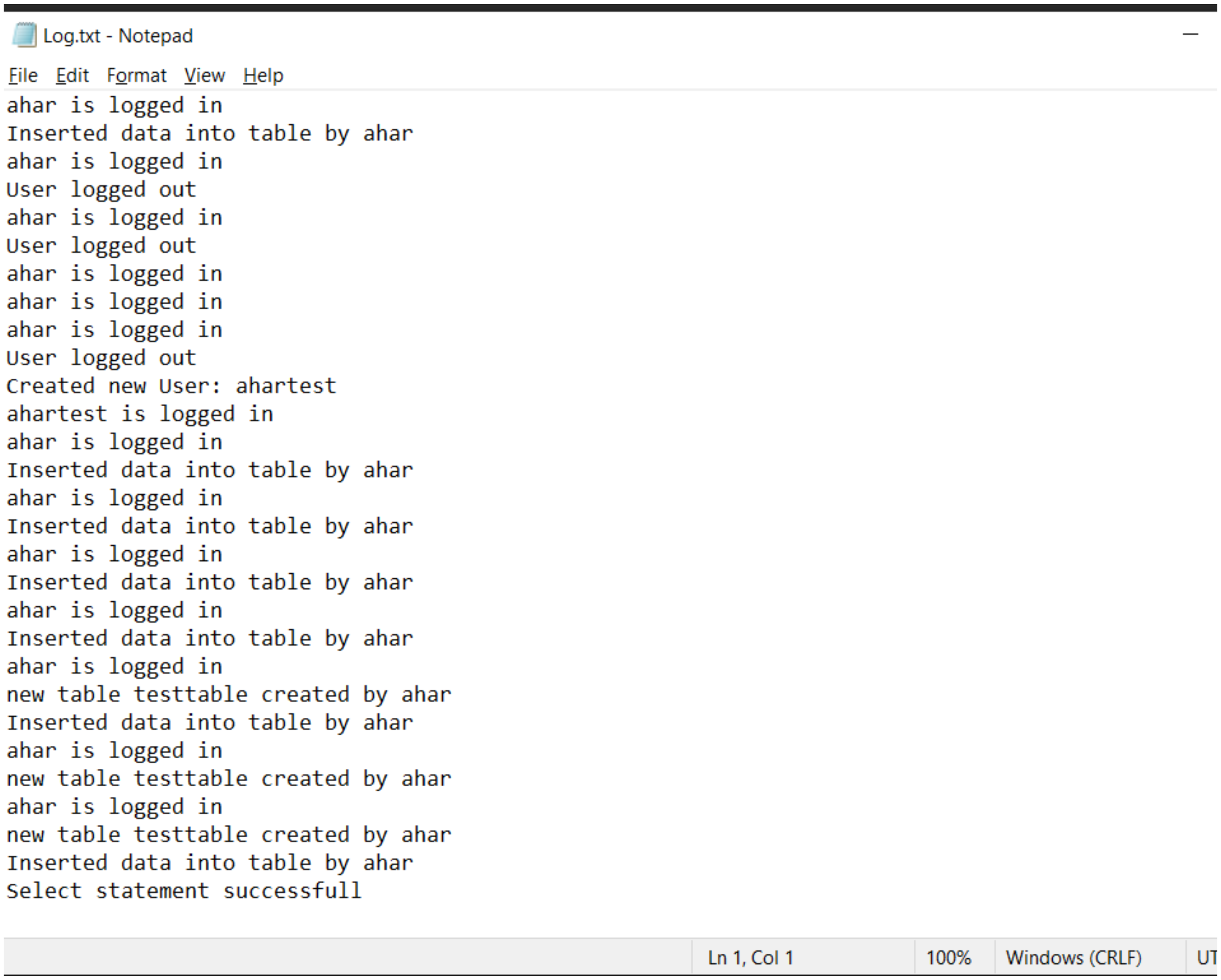
If the user chooses to add a user, they are prompted to enter a username and password. The password is then encrypted using the encryptPassword method and stored in the file. If the user chooses to login, they are prompted to enter a username and password. The program then calls the checkAuthentication method to validate the provided credentials. If the authentication is successful, it displays a success message and proceeds to perform operations related to the logged-in user. If the authentication fails, an error message is displayed.



```
"C:\Program Files\Java\jdk-17\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2022.3.2\lib\idea_rt.jar=64778:C:\Program Files\JetB
 2022.3.2\bin" -Dfile.encoding=UTF-8 -classpath "D:\#Sem 2 Summer\DATA\Assignment 2\Solution\Datagitlab\Assignment2\target\classes" org.example.DBMS
===================
Welcome to the DBMS
===================

1. Add a user
2. Login
3. Exit
Enter your choice: 2
Enter Username :
ahar
Enter Password :
abc
Successfully logged in


1. Add a user
2. Login
3. Exit
Enter your choice: 3
Program exited Successfuly

Process finished with exit code 0
```

Figure 5: Successful User Login

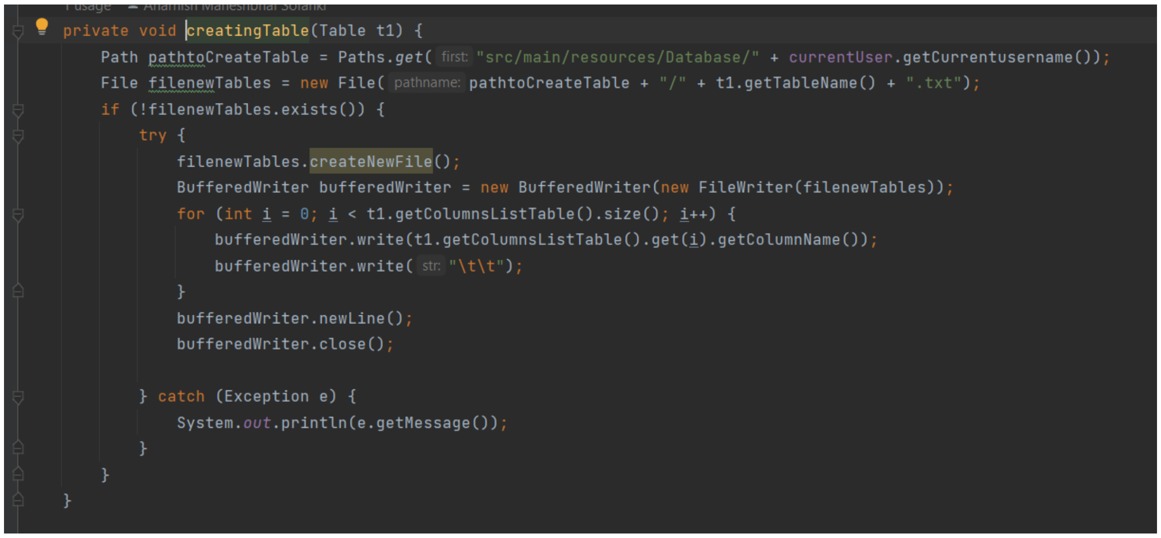
Throughout the program, the Log class is used to write log messages for various events, such as creating a new user, logging in, and inserting data into a table.



```
ahar is logged in
Inserted data into table by ahar
ahar is logged in
User logged out
ahar is logged in
User logged out
ahar is logged in
ahar is logged in
ahar is logged in
User logged out
Created new User: ahartest
ahartest is logged in
ahar is logged in
Inserted data into table by ahar
ahar is logged in
Inserted data into table by ahar
ahar is logged in
Inserted data into table by ahar
ahar is logged in
Inserted data into table by ahar
ahar is logged in
new table testtable created by ahar
Inserted data into table by ahar
ahar is logged in
new table testtable created by ahar
ahar is logged in
new table testtable created by ahar
Inserted data into table by ahar
Select statement successfull
```

Figure 6: User Logs in the text file

- Implementation of Queries (DDL & DML):
  - **Create table query**: main business logic happens in creatingTable() of Userservice class.



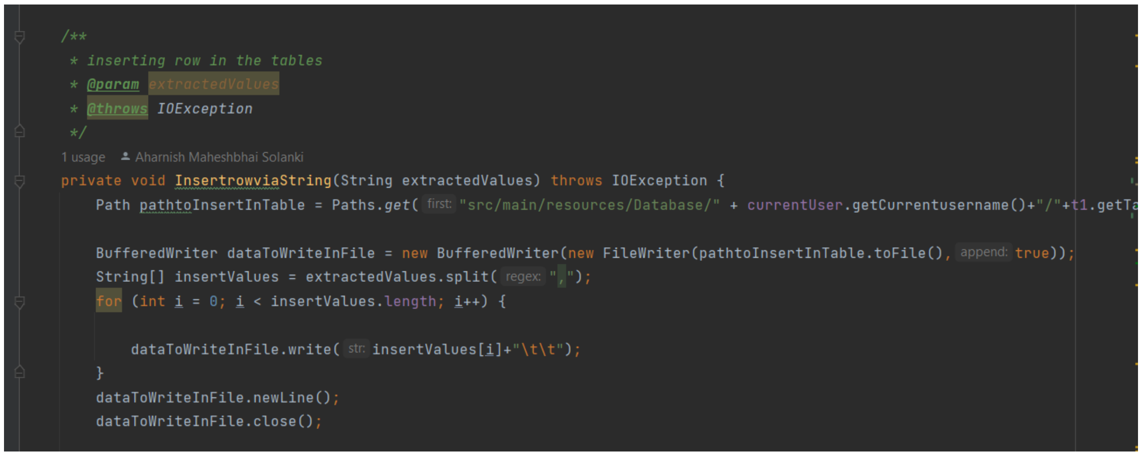
```java
private void creatingTable(Table t1) {
    Path pathtoCreateTable = Paths.get( first: "src/main/resources/Database/" + currentUser.getCurrentusername());
    File filenewTables = new File( pathname: pathtoCreateTable + "/" + t1.getTableName() + ".txt");
    if (!filenewTables.exists()) {
        try {
            filenewTables.createNewFile();
            BufferedWriter bufferedWriter = new BufferedWriter(new FileWriter(filenewTables));
            for (int i = 0; i < t1.getColumnsListTable().size(); i++) {
                bufferedWriter.write(t1.getColumnsListTable().get(i).getColumnName());
                bufferedWriter.write( str: "\t\t");
            }
            bufferedWriter.newLine();
            bufferedWriter.close();

        } catch (Exception e) {
            System.out.println(e.getMessage());
        }
    }
}
```

  -

- There is a Table class which has column entity and String variable tablename.
- The column class holds the information of metadata of table class.

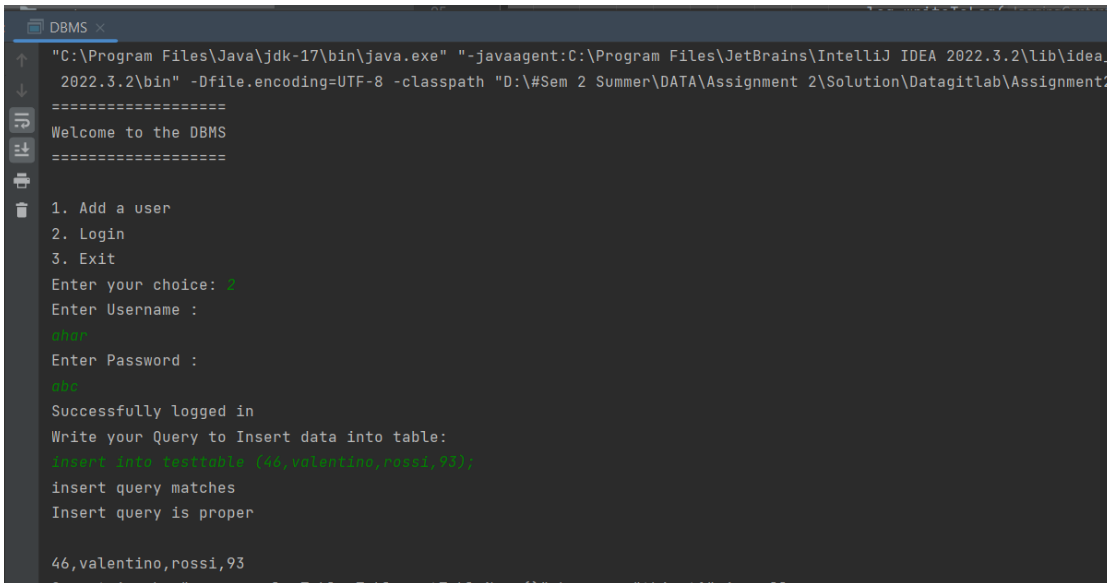
➢ **Insert Query:**

- Insert query worked on table. InsertrowviaString() adds the new data in the txt tables files.



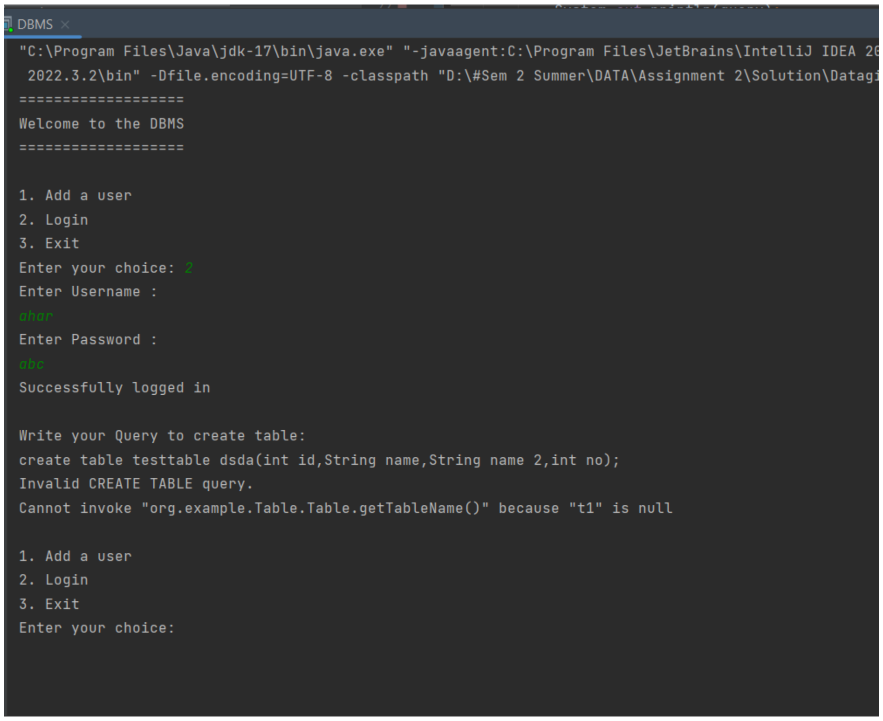
```
/**
 * inserting row in the tables
 * @param extractedValues
 * @throws IOException
 */
1 usage   Aharnish Maheshbhai Solanki
private void InsertrowviaString(String extractedValues) throws IOException {
    Path pathtoInsertInTable = Paths.get( first: "src/main/resources/Database/" + currentUser.getCurrentusername()+"/"+t1.getTa

    BufferedWriter dataToWriteInFile = new BufferedWriter(new FileWriter(pathtoInsertInTable.toFile(), append: true));
    String[] insertValues = extractedValues.split( regex: ",");
    for (int i = 0; i < insertValues.length; i++) {

        dataToWriteInFile.write( str: insertValues[i]+"\t\t");
    }
    dataToWriteInFile.newLine();
    dataToWriteInFile.close();
```

- **Fig**



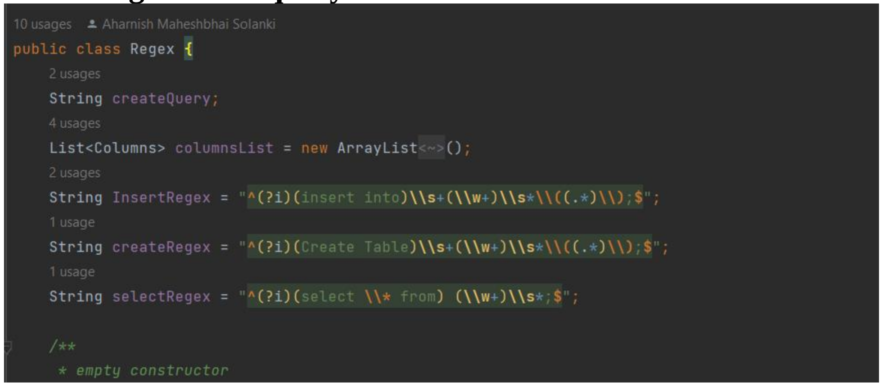
```
"C:\Program Files\Java\jdk-17\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2022.3.2\lib\idea_
 2022.3.2\bin" -Dfile.encoding=UTF-8 -classpath "D:\#Sem 2 Summer\DATA\Assignment 2\Solution\Datagitlab\Assignment2
===================
Welcome to the DBMS
===================

1. Add a user
2. Login
3. Exit
Enter your choice: 2
Enter Username :
ahar
Enter Password :
abc
Successfully logged in
Write your Query to Insert data into table:
insert into testtable (46,valentino,rossi,93);
insert query matches
Insert query is proper

46,valentino,rossi,93
```

- **Fig**



```
DBMS
"C:\Program Files\Java\jdk-17\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 20
 2022.3.2\bin" -Dfile.encoding=UTF-8 -classpath "D:\#Sem 2 Summer\DATA\Assignment 2\Solution\Datagi
===================
Welcome to the DBMS
===================

1. Add a user
2. Login
3. Exit
Enter your choice: 2
Enter Username :
ahar
Enter Password :
abc
Successfully logged in

Write your Query to create table:
create table testtable dsda(int id,String name,String name 2,int no);
Invalid CREATE TABLE query.
Cannot invoke "org.example.Table.Table.getTableName()" because "t1" is null

1. Add a user
2. Login
3. Exit
Enter your choice:
```

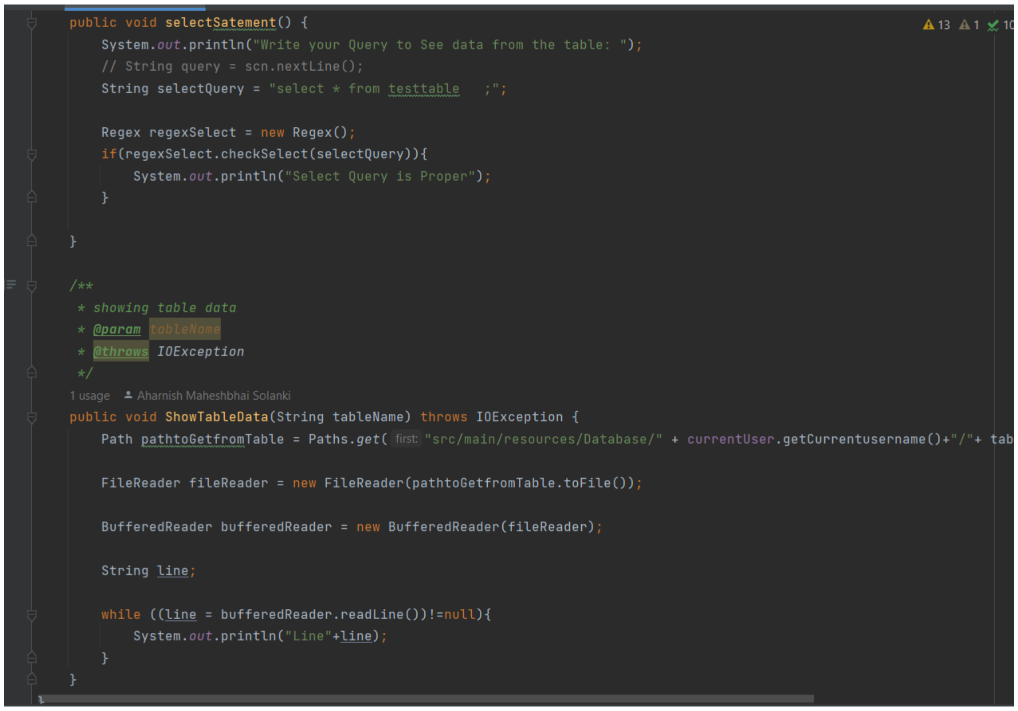
- ○
    - ▪ **Fig: invalid query**



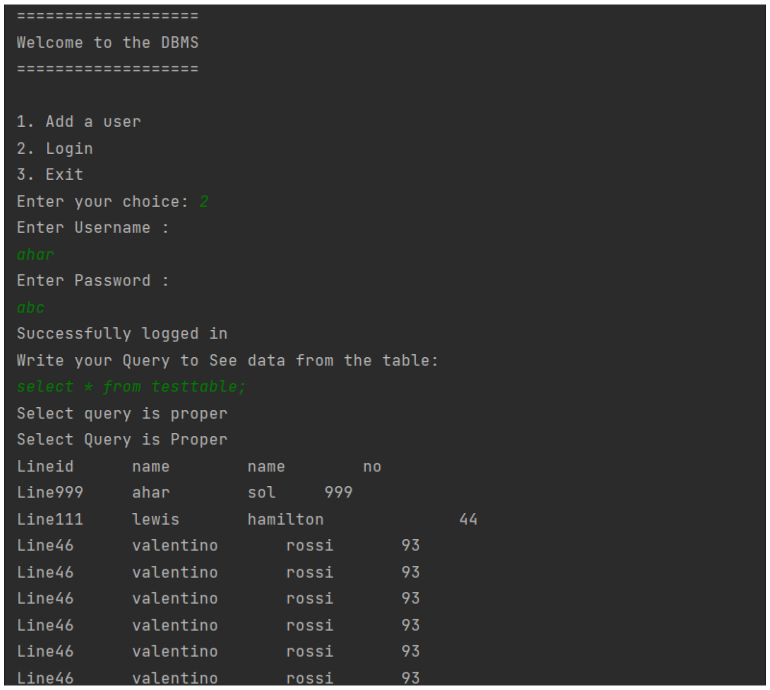
```
10 usages  Aharnish Maheshbhai Solanki
public class Regex {
    2 usages
    String createQuery;
    4 usages
    List<Columns> columnsList = new ArrayList<~>();
    2 usages
    String InsertRegex = "^(?i)(insert into)\\s+(\\w+)\\s*\\((.*)\\);$";
    1 usage
    String createRegex = "^(?i)(Create Table)\\s+(\\w+)\\s*\\((.*)\\);$";
    1 usage
    String selectRegex = "^(?i)(select \\* from) (\\w+)\\s*;$";

    /**
     * empty constructor
```

- ○
    - ▪ **Fig : regex used for create table**

- **Select query:**
    - Select query to get the data from the txt files of the table
    - The select statement in the Userservice class shows the data from table selected.

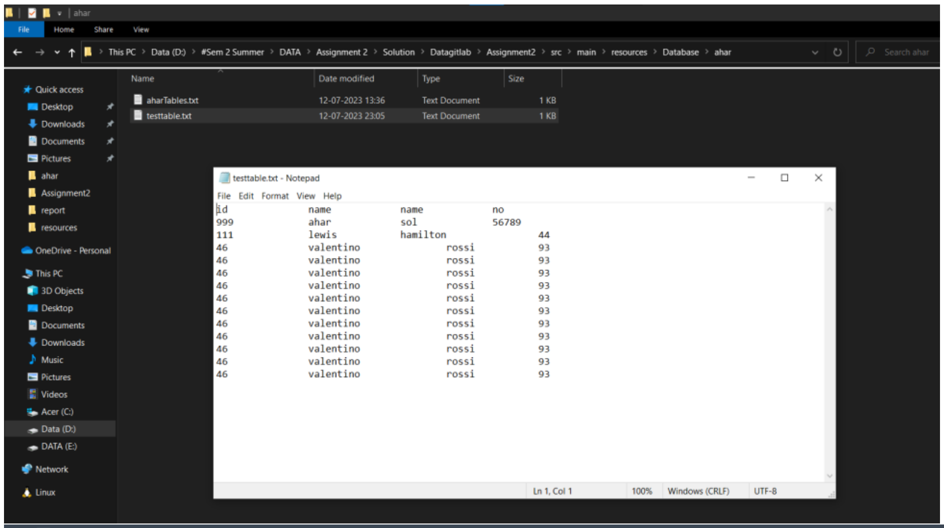


```
public void selectSatement() {
    System.out.println("Write your Query to See data from the table: ");
    // String query = scn.nextLine();
    String selectQuery = "select * from testtable   ;";

    Regex regexSelect = new Regex();
    if(regexSelect.checkSelect(selectQuery)){
        System.out.println("Select Query is Proper");
    }

}

/**
 * showing table data
 * @param tableName
 * @throws IOException
 */
1 usage   Aharnish Maheshbhai Solanki
public void ShowTableData(String tableName) throws IOException {
    Path pathtoGetfromTable = Paths.get( first: "src/main/resources/Database/" + currentUser.getCurrentusername()+"/"+ tab

    FileReader fileReader = new FileReader(pathtoGetfromTable.toFile());

    BufferedReader bufferedReader = new BufferedReader(fileReader);

    String line;

    while ((line = bufferedReader.readLine())!=null){
        System.out.println("Line"+line);
    }
}
```



```
===================
Welcome to the DBMS
===================

1. Add a user
2. Login
3. Exit
Enter your choice: 2
Enter Username :
ahar
Enter Password :
abc
Successfully logged in
Write your Query to See data from the table:
select * from testtable;
Select query is proper
Select Query is Proper
Lineid      name        name        no
Line999     ahar        sol     999
Line111     lewis       hamilton              44
Line46      valentino       rossi       93
Line46      valentino       rossi       93
Line46      valentino       rossi       93
Line46      valentino       rossi       93
Line46      valentino       rossi       93
Line46      valentino       rossi       93
```

➢ Update Query :



ahar

This PC › Data (D:) › #Sem 2 Summer › DATA › Assignment 2 › Solution › Datagitlab › Assignment2 › src › main › resources › Database › ahar

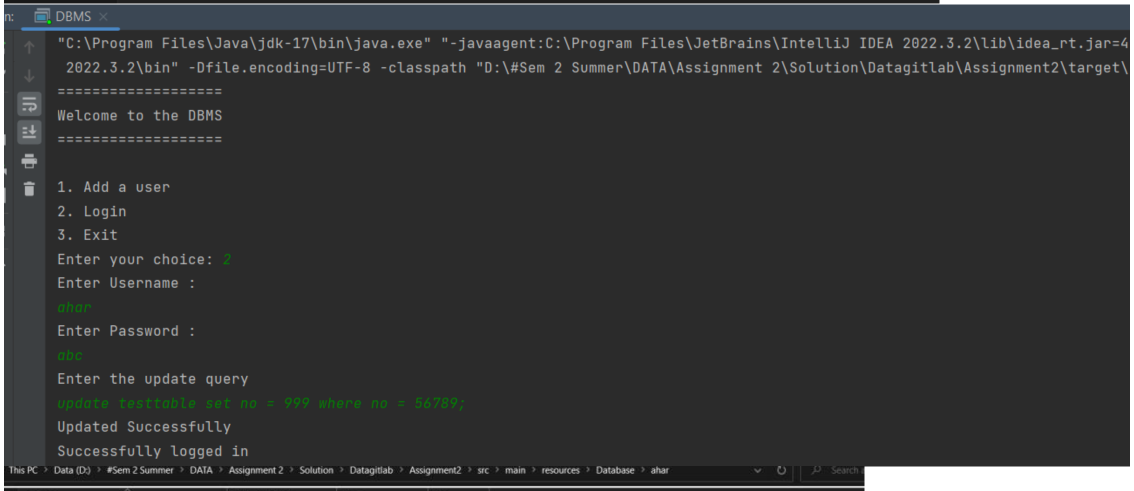
Name	Date modified	Type	Size
aharTables.txt	12-07-2023 13:36	Text Document	1 KB
testtable.txt	12-07-2023 23:05	Text Document	1 KB

testtable.txt - Notepad

```
id              name            name            no
999             ahar            sol             56789
111             lewis           hamilton                44
46              valentino               rossi           93
46              valentino               rossi           93
46              valentino               rossi           93
46              valentino               rossi           93
46              valentino               rossi           93
46              valentino               rossi           93
46              valentino               rossi           93
46              valentino               rossi           93
46              valentino               rossi           93
46              valentino               rossi           93
46              valentino               rossi           93
```

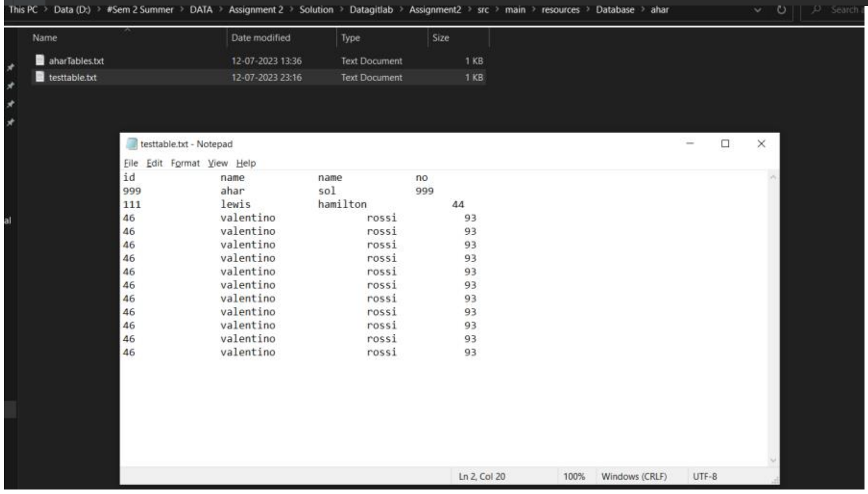
Ln 1, Col 1 | 100% | Windows (CRLF) | UTF-8

➢



```
"C:\Program Files\Java\jdk-17\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2022.3.2\lib\idea_rt.jar=4
 2022.3.2\bin" -Dfile.encoding=UTF-8 -classpath "D:\#Sem 2 Summer\DATA\Assignment 2\Solution\Datagitlab\Assignment2\target\
===================
Welcome to the DBMS
===================

1. Add a user
2. Login
3. Exit
Enter your choice: 2
Enter Username :
ahar
Enter Password :
abc
Enter the update query
update testtable set no = 999 where no = 56789;
Updated Successfully
Successfully logged in
```

➢



This PC › Data (D:) › #Sem 2 Summer › DATA › Assignment 2 › Solution › Datagitlab › Assignment2 › src › main › resources › Database › ahar

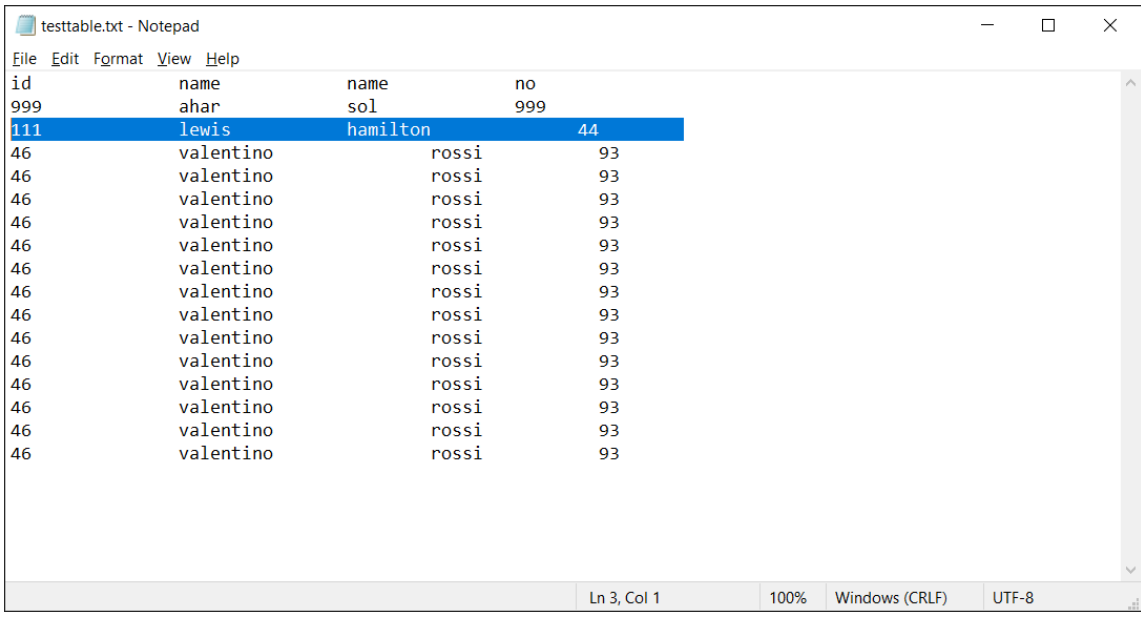
Name	Date modified	Type	Size
aharTables.txt	12-07-2023 13:36	Text Document	1 KB
testtable.txt	12-07-2023 23:16	Text Document	1 KB

testtable.txt - Notepad

```
id              name            name            no
999             ahar            sol             999
111             lewis           hamilton                44
46              valentino               rossi           93
46              valentino               rossi           93
46              valentino               rossi           93
46              valentino               rossi           93
46              valentino               rossi           93
46              valentino               rossi           93
46              valentino               rossi           93
46              valentino               rossi           93
46              valentino               rossi           93
46              valentino               rossi           93
46              valentino               rossi           93
```

Ln 2, Col 20 | 100% | Windows (CRLF) | UTF-8

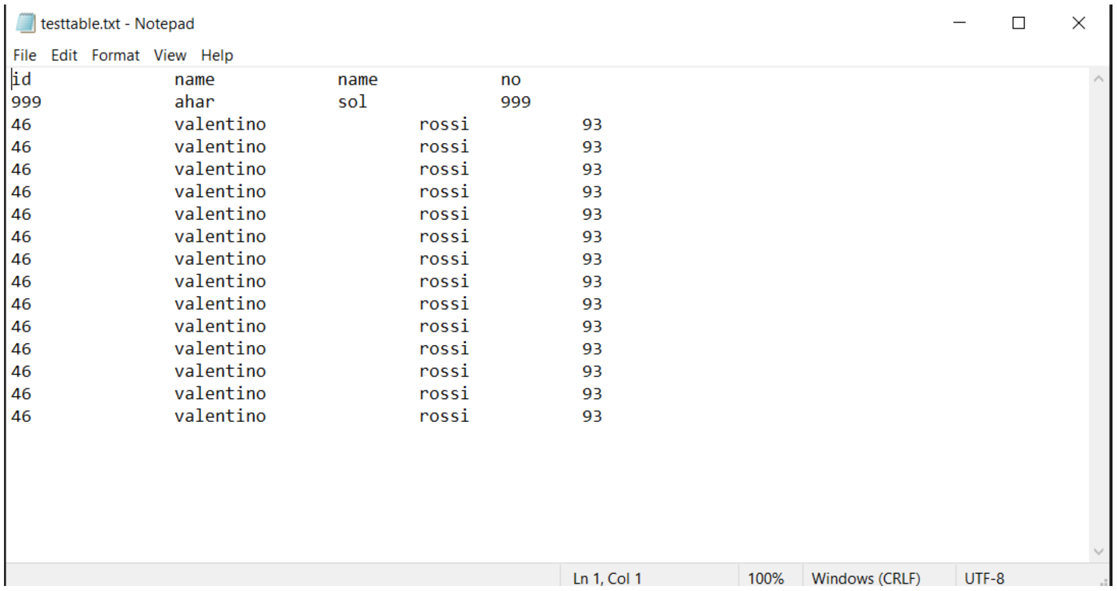
➢ Delete Query :


testtable.txt - Notepad

```
id              name            name            no
999             ahar            sol             999
111             lewis           hamilton              44
46              valentino               rossi           93
46              valentino               rossi           93
46              valentino               rossi           93
46              valentino               rossi           93
46              valentino               rossi           93
46              valentino               rossi           93
46              valentino               rossi           93
46              valentino               rossi           93
46              valentino               rossi           93
46              valentino               rossi           93
46              valentino               rossi           93
46              valentino               rossi           93
46              valentino               rossi           93
46              valentino               rossi           93
```

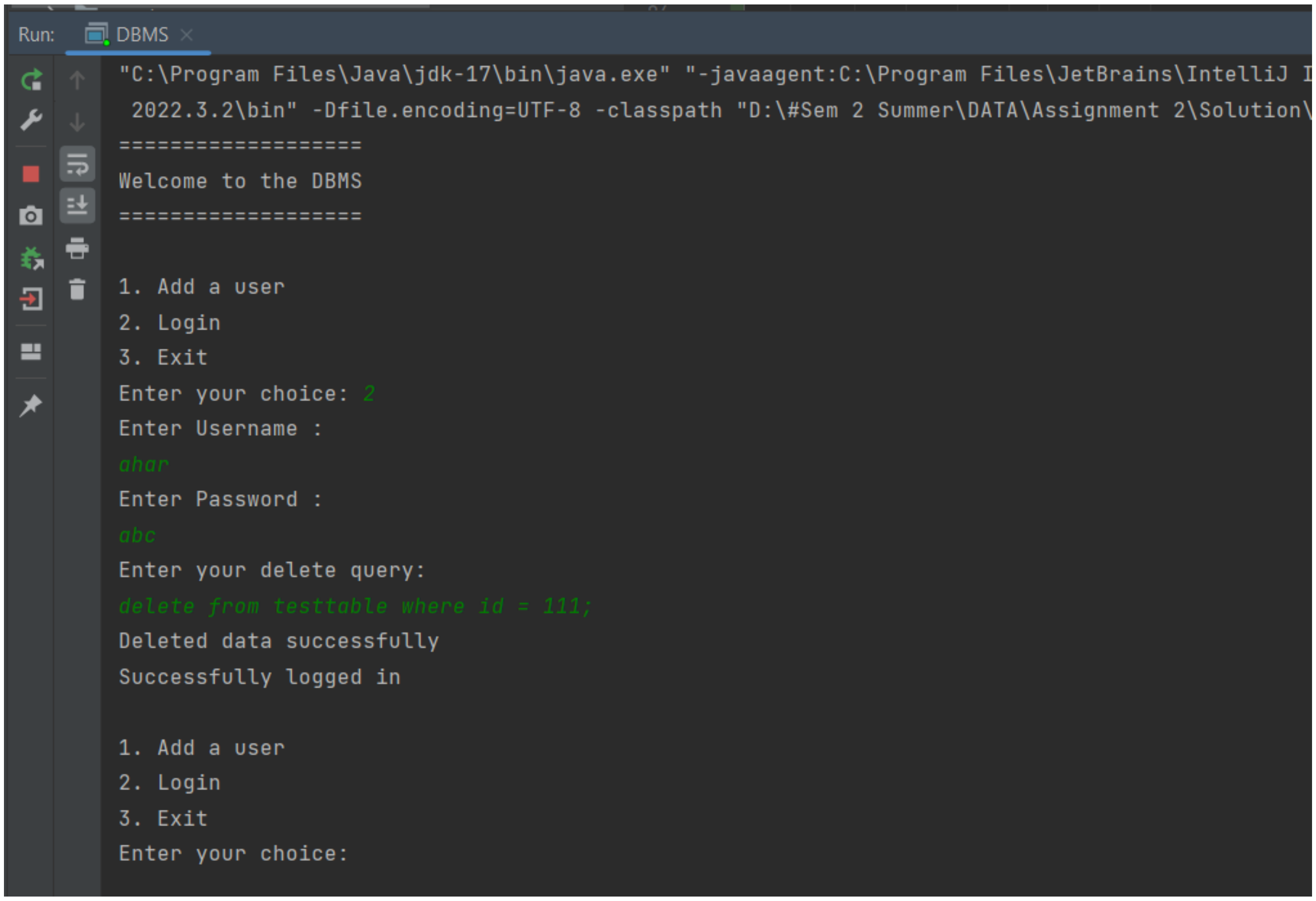
Ln 3, Col 1 100% Windows (CRLF) UTF-8

➤


testtable.txt - Notepad

```
id              name            name            no
999             ahar            sol             999
46              valentino               rossi           93
46              valentino               rossi           93
46              valentino               rossi           93
46              valentino               rossi           93
46              valentino               rossi           93
46              valentino               rossi           93
46              valentino               rossi           93
46              valentino               rossi           93
46              valentino               rossi           93
46              valentino               rossi           93
46              valentino               rossi           93
46              valentino               rossi           93
46              valentino               rossi           93
46              valentino               rossi           93
```

Ln 1, Col 1 100% Windows (CRLF) UTF-8



```
"C:\Program Files\Java\jdk-17\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ I
 2022.3.2\bin" -Dfile.encoding=UTF-8 -classpath "D:\#Sem 2 Summer\DATA\Assignment 2\Solution\
===================
Welcome to the DBMS
===================

1. Add a user
2. Login
3. Exit
Enter your choice: 2
Enter Username :
ahar
Enter Password :
abc
Enter your delete query:
delete from testtable where id = 111;
Deleted data successfully
Successfully logged in

1. Add a user
2. Login
3. Exit
Enter your choice:
```

References:

[1] "The Eclipse Foundation, Eclipse Downloads," *The Eclipse Foundation*. [Online]. Available: https://www.eclipse.org/downloads/ [Accessed: 08 July 2023].

[2] "Javadoc coding standards,". [Online]. Available: https://blog.joda.org/2012/11/javadoc-coding-standards.html [Accessed: 12 July 2023].

[3] L. Gupta, "Java hashing using MD5, Sha, PBKDF2, Bcrypt and scrypt, HowToDoInJava." [Online] Available: https://howtodoinjava.com/java/java-security/how-to-generate-secure-password-hash-md5-sha-pbkdf2-bcrypt-examples/ [Accessed: 08 July 2023].