

CSCI 5408 - DATA MANAGEMENT AND WAREHOUSING
ASSIGNMENT – 2 REPORT
GOWRI PRASHANTH KANAGARAJ- B00942544

Problem 1:

- **Io#1:** Perform research on distributed data management and security – To achieve this task, you need to read and understand a given published research material and write a summary of your understandings.

Summary - “A Comparative Analysis of Data Fragmentation in Distributed Database” [1]

The research paper titled "A Comparative Analysis of Data Fragmentation in Distributed Database" provides an overview of Fragmentation of data in distributed database systems [1]. The paper investigates the distributed database environment, fragmentations, and distributed database design. It compares three types of fragmentation: horizontal, vertical, and mixed fragmentation. It evaluates fragmentation strategies in distributed databases using correctness rules to suggest the best approach for effective database design.

The paper introduces the significance of distributed database technology in data processing and discusses their fundamental concepts. It interprets that distributed database is a single logical database spread physically across computers connected by a data communications network. The distribution of data involves fragmentation, replication, and allocation processes. The distributed database design process is outlined, consisting of three phases: initial design, redesign, and materialization. The initial design involves fragmentation and allocation algorithms aimed at minimizing transaction processing costs. The redesign phase focuses on generating new fragmentation and allocation schemes based on changes in the distributed database environment. Materialization involves implementing the new fragmentation and allocation scheme.

In the next section, the paper reviews related works that concern fragmentation in distributed database systems. It discussing various researchers' approaches and algorithms for fragmentation schemes and allocation schemes. The reviewed research works cover vertical fragmentation, horizontal fragmentation, data allocation, replication, and optimization of communication costs.

In the following section, it talks in depth about Fragmentation. It states that fragmentation in distributed databases aims to improve several aspects and explains about them. To determine the appropriate fragmentation strategy, the paper highlights the use of both quantitative and qualitative information. The advantages and disadvantages of fragmentation are pointed out. Horizontal fragmentation, which partitions a relation into disjoint tuples or instances, and vertical

fragmentation, which divides a relation into separate sets of columns or attributes, are analyzed. Mixed fragmentation, a combination of horizontal and vertical techniques, is also discussed. It provides illustrations of the different fragmentation types.

Subsequently, in the next segment, data fragmentation for the different fragment types are explained with real world examples and illustrations.

It then introduces correctness rules of fragmentation, including completeness, reconstruction, and dis-jointness. These rules ensure that the fragmented data can be reconstructed and that each data item appears in the appropriate fragment. The correctness rules for all the fragmentation strategies are illustrated in a tabular format.

In conclusion, the paper offers valuable insights into data fragmentation in distributed database systems. It presents a comparative analysis of fragmentation strategies and provides a useful resource for researchers and practitioners in the field of distributed databases.

Scope of Improvements

The paper provides a solid foundation for the topic of data fragmentation, however there can be scope of improvements in terms of technical details and concepts. Improved fragmentation strategies and algorithms can be researched and analyzed. For example., to improve the performance of distributed database systems, Query processing optimization techniques are an important concern [2]. Vertical fragmentation by the K-means classification algorithm can be a better solution for this problem [2]. k-means approach for vertical fragmentation and allocation can also be used to significantly reduce the costs associated with data transmission in distribution [3]. Apart from this, challenges and limitations of fragmentation in distributed databases can be further explored. Tables in distributed database systems are fragmented and replicated across sites to minimize network communication costs. These challenges are traditionally addressed through static fragmentation techniques [4]. This method degrades performance which can be overcome by dynamic fragmentation and replica management [4]. It successfully minimizes communication costs for common access patterns, indicating the practicality of the dynamic fragmentation method [4].

References:

- [1] "A comparative analysis of data fragmentation in distributed database," *ieeexplore*, [Online]. Available: <https://ieeexplore-ieee-org.ezproxy.library.dal.ca/document/8079934> [Accessed: July 04,2023]

- [2] "Vertical fragmentation and allocation in distributed databases using k-mean algorithm," *ProQuest*, [Online]. Available: <https://www.proquest.com/docview/2785266432?fromopenview=true&pq-origsite=gscholar> [Accessed: July 04,2023]

- [3] "On K-means clustering-based approach for DDBSs design," *journalofbigdata*, [Online]. Available: <https://journalofbigdata.springeropen.com/articles/10.1186/s40537-020-00306-9> [Accessed: July 04,2023]

- [4] "DYFRAM: Dynamic fragmentation and replica management in distributed database systems," *researchgate*, [Online]. Available: <https://journalofbigdata.springeropen.com/articles/10.1186/s40537-020-00306-9> [Accessed: July 04,2023]

Problem 2:

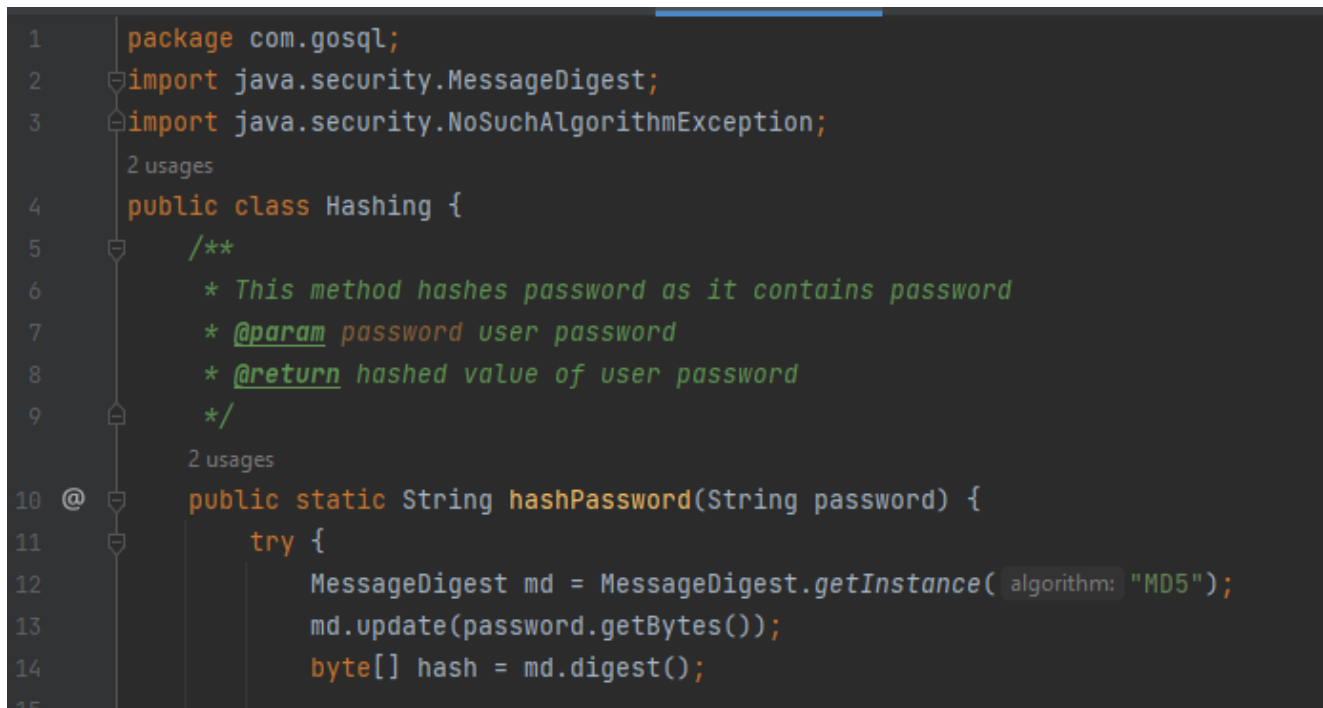
- **Io#2:** Build a prototype of a light-weight DBMS using Java programming language, which performs at least 3 required functions, and 1 optional function from a given list of DBMS functions. If your application performs both optional functions of DBMS, then you may get additional points for novelty. **[Note:** Your code must be written by you, and will be checked for academic integrity]

GITLAB repo: https://git.cs.dal.ca/kanagaraj/csci5408_s23_b00942544_gowriprashanth_kanagaraj

1. Java IDE used : IntelliJ

JDK version : OpenJDK 20

2. Followed JavaDocs specification for commenting styles like @param, @return have been used throughout the code.



```
1 package com.gosql;
2 import java.security.MessageDigest;
3 import java.security.NoSuchAlgorithmException;
4
5 public class Hashing {
6     /**
7      * This method hashes password as it contains password
8      * @param password user password
9      * @return hashed value of user password
10    */
11    public static String hashPassword(String password) {
12        try {
13            MessageDigest md = MessageDigest.getInstance("MD5");
14            md.update(password.getBytes());
15            byte[] hash = md.digest();
```

Fig 1: JavaDocs specification for commenting styles

3. SOLID Principles:

Followed SOLID Principles while creating the application:

Single Responsibility Principle:

According to Single Responsibility Principle, a class should have one reason change [1]. Every class in application has only one reason/purpose to change. This allows to clearly differentiate between the responsibilities of each class.

Open Closed Principle:

According to Open Closed Principle, software entities (classes, modules, functions, and so on) should be open for extension, but closed for modification [2]. Classes and modules in this application follows OCP. Thus the developers can easily add a new behavior by extension rather than modification of the code.

Liskov Substitution Principle:

According to Liskov Substitution Principle, objects of a superclass should be able to be replaced with objects of a subclass without affecting the correctness of the program [3]. The application follows LSP, thus it does not have any Guard clauses and Run Time Type Identification.

Interface Segregation Principle:

According to Interface Segregation Principle No client should be forced to depend on methods it does not use [4]. The application follows ISP. Thus None of the clients are forced to use any of the additional functionalities

Dependency Inversion Principle:

According to Dependency Inversion Principle, high level modules should not depend on low level modules; both should depend on abstractions [5]. The application follows DIP, thus no high-level modules depend on low level modules.

4. Application is console based and accepts user input in the form SQL query after user has logged in.

As shown below, the application accepts user query as an input and provides the respective output from the command line.

```
Enter username: Gowri
Enter password: prash
Question:
color
Enter answer: blue
Login successful! Welcome, Gowri!
Please press 1 to enter query or press 0 to exit:
1
Please enter the query:
Select * FROM books;
query:Select * FROM books
bookId bookName pages
1 Aiwin 50
2 Alchemist 200
7 Two states 300
3 Alchemist 350
4 chemist 250
6 mist 130
8 newbook 0
9 Two states 300
10 newbook1 10
8 newbook 0
9 Two states 300
```

Fig 2: Query input and console output

The application allows users to create databases using query as an additional feature.

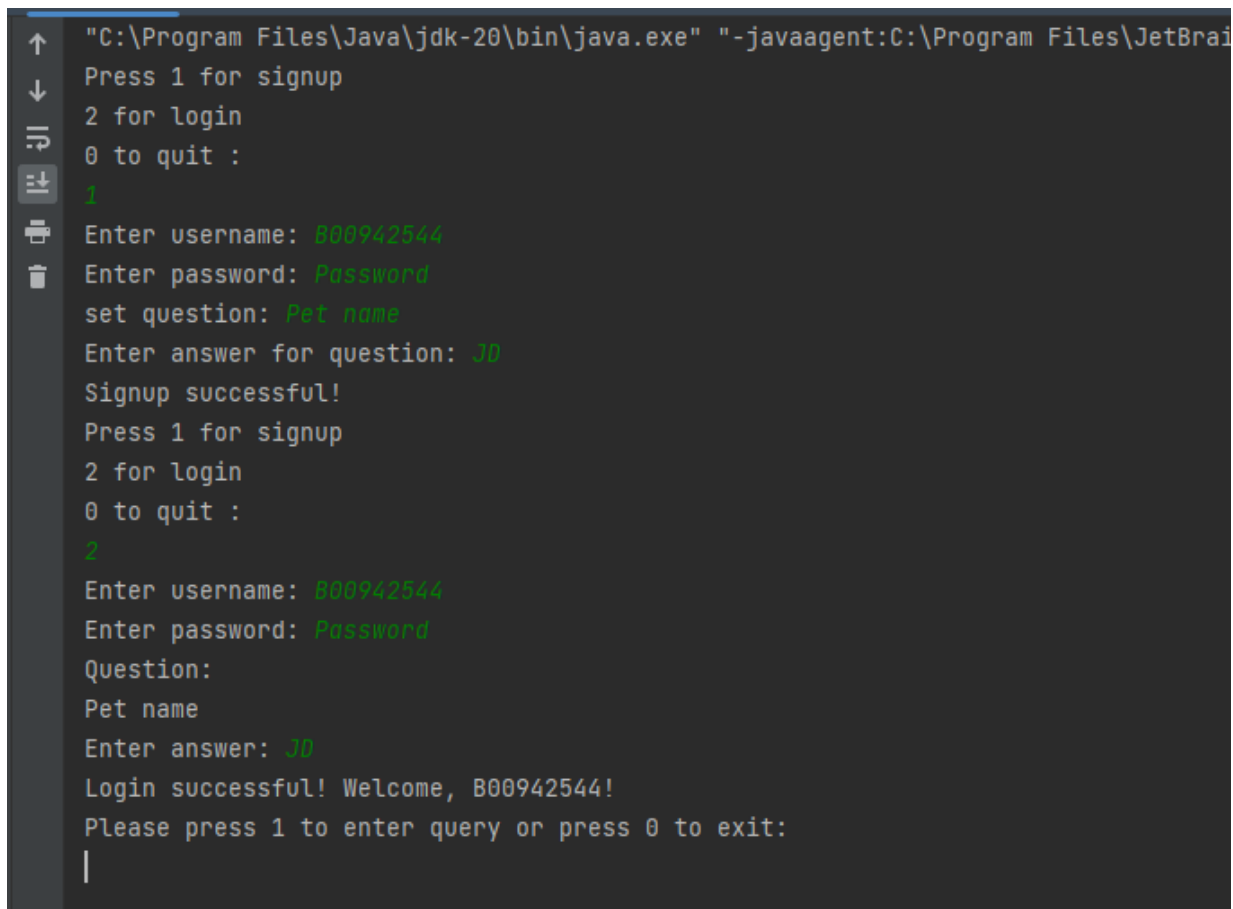
CREATE DATABASE database_name;

```
Please press 1 to enter query or press 0 to exit:
1
Please enter the query:
CREATE DATABASE goDB;
query:CREATE DATABASE goDB
Created Database:true
Please press 1 to enter query or press 0 to exit:
```

Fig 3: User database creation

5.Two Factor Authentication:

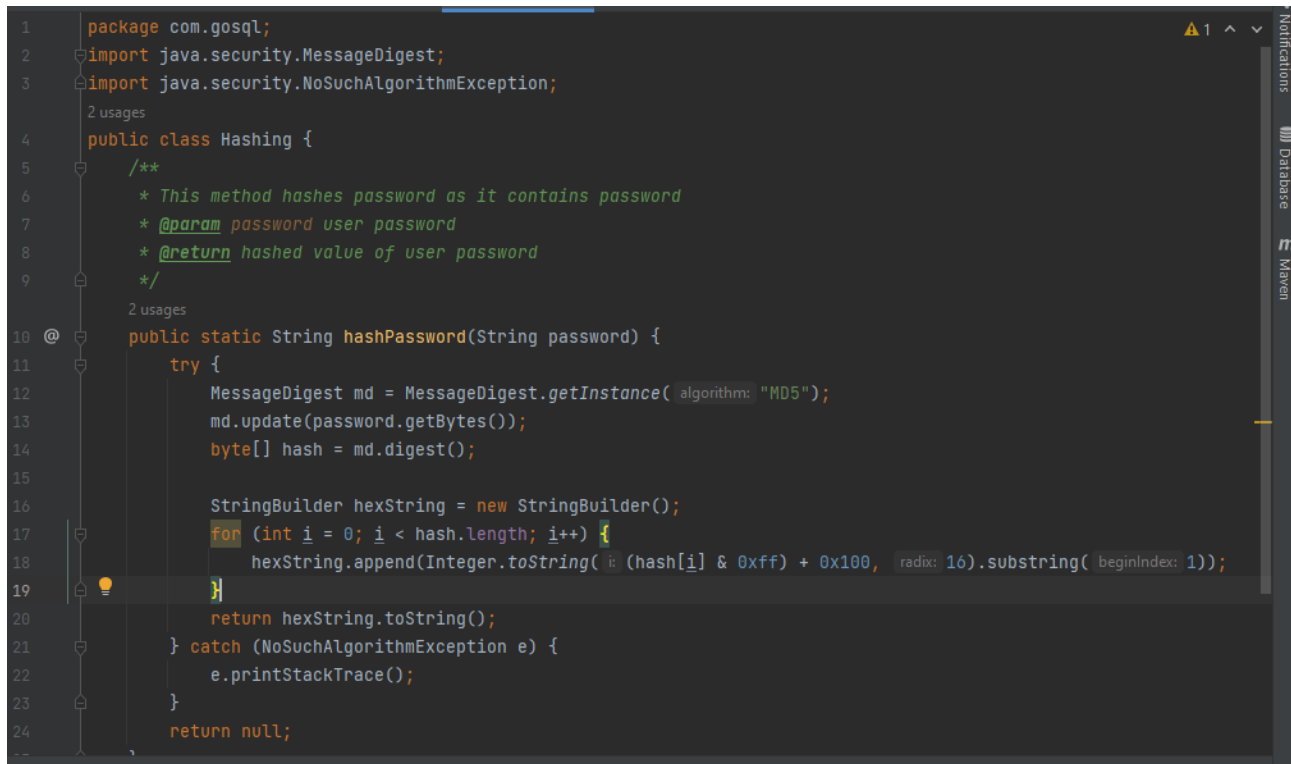
- The application needs users to setup username, password, question and answer while Signing up.
- After successful signup, the user is able to login using the username, password and answer for the set question.
- The application supports multiple users, as multiple users can sign up and set up their account.



```
"C:\Program Files\Java\jdk-20\bin\java.exe" "-javaagent:C:\Program Files\JetBrai
Press 1 for signup
2 for login
0 to quit :
1
Enter username: B00942544
Enter password: Password
set question: Pet name
Enter answer for question: JD
Signup successful!
Press 1 for signup
2 for login
0 to quit :
2
Enter username: B00942544
Enter password: Password
Question:
Pet name
Enter answer: JD
Login successful! Welcome, B00942544!
Please press 1 to enter query or press 0 to exit:
|
```

Fig 4: Two factor authentication

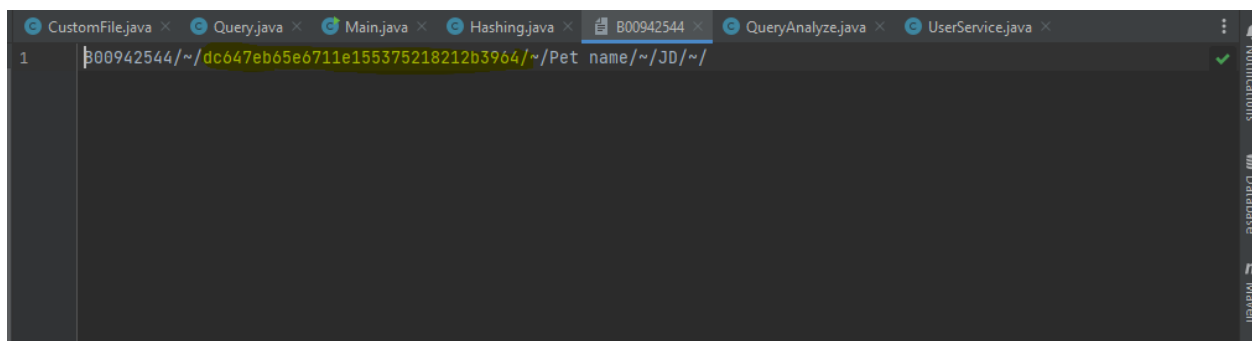
For hashing, md5 a standard java library is used. [6]



```
1 package com.gosql;
2 import java.security.MessageDigest;
3 import java.security.NoSuchAlgorithmException;
4 public class Hashing {
5     /**
6      * This method hashes password as it contains password
7      * @param password user password
8      * @return hashed value of user password
9      */
10    public static String hashPassword(String password) {
11        try {
12            MessageDigest md = MessageDigest.getInstance("MD5");
13            md.update(password.getBytes());
14            byte[] hash = md.digest();
15
16            StringBuilder hexString = new StringBuilder();
17            for (int i = 0; i < hash.length; i++) {
18                hexString.append(Integer.toString((hash[i] & 0xff) + 0x100, 16).substring(1));
19            }
20            return hexString.toString();
21        } catch (NoSuchAlgorithmException e) {
22            e.printStackTrace();
23        }
24        return null;
25    }
```

Fig 5: Hashing technique using md5 library

The password is hashed as it is a sensitive information and can be easily fetched to access the user account. The hashed value is stored in user information file as follows:



```
1 B00942544/~ /dc647eb65e6711e155375218212b3964/~ /Pet name/~ /JD/~ /
```

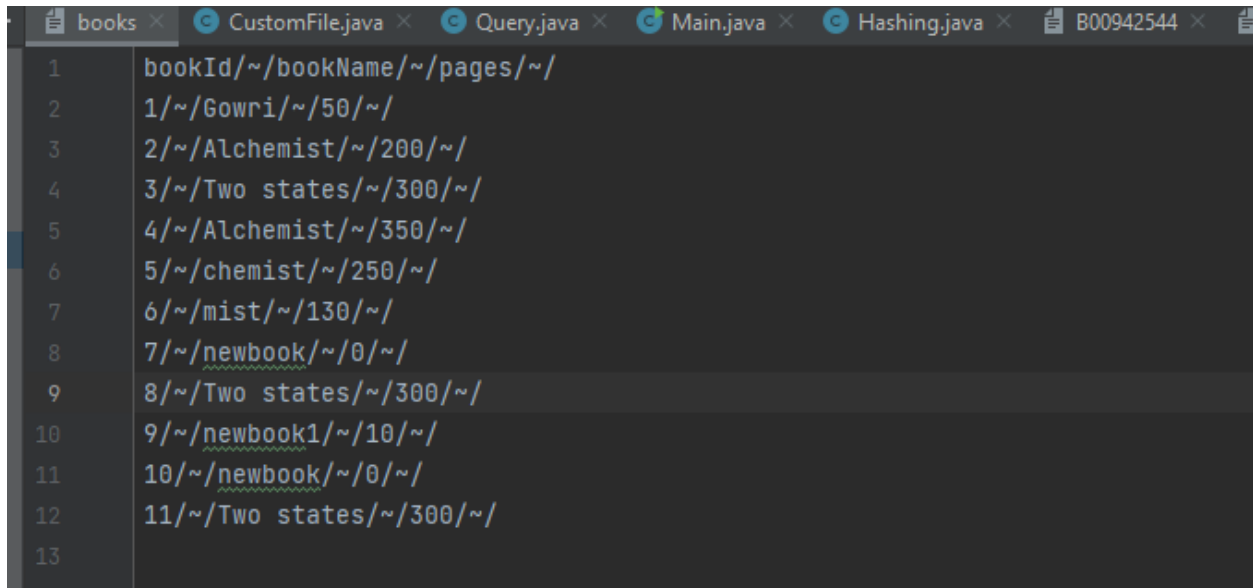
Fig 6: Hashed data stored in file

The stored value is retrieved. The password entered by user while login is hashed and matched with retrieved value. If the values match, user is able to login.

6.Persistent storage:

First, user account information such as username, password, question and answer are stored in file designed with delimiter `/~/` as showed in image 5.

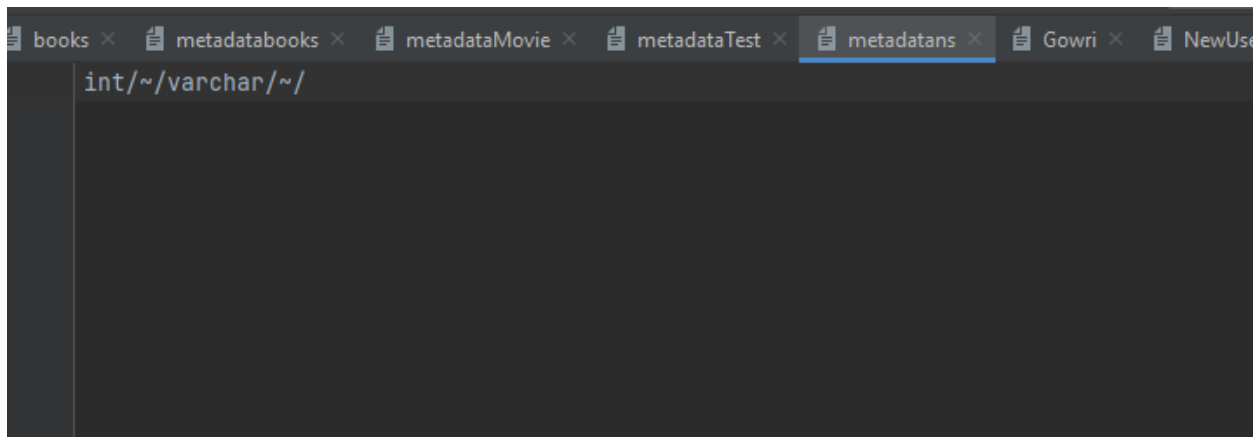
The user table data such as column names and their respective data are stored in table file: The delimiter `/~/` is used



```
1 bookId/~/bookName/~/pages/~/
2 1/~/Gowri/~/50/~/
3 2/~/Alchemist/~/200/~/
4 3/~/Two states/~/300/~/
5 4/~/Alchemist/~/350/~/
6 5/~/chemist/~/250/~/
7 6/~/mist/~/130/~/
8 7/~/newbook/~/0/~/
9 8/~/Two states/~/300/~/
10 9/~/newbook1/~/10/~/
11 10/~/newbook/~/0/~/
12 11/~/Two states/~/300/~/
13
```

Fig 7: data stored in files with delimiters

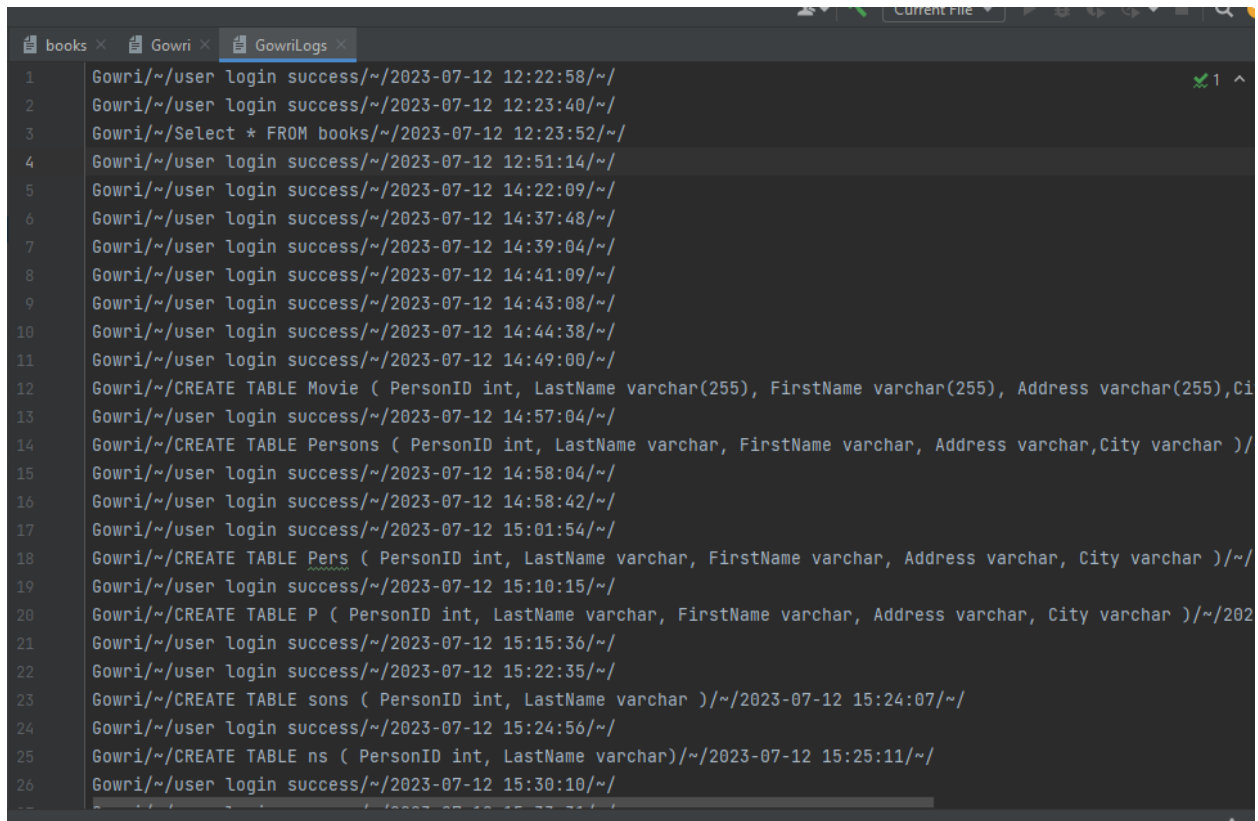
The meta-data for the table like data type is stored in the meta-data file. The delimiter `/~/` is used



```
int/~/varchar/~/
```

Fig 8: meta-data stored in files with delimiters

Logs are also saved to log files. Logging helps in tracing out errors while running the application. The delimiter `/~/` is used.



```
1 Gowri/~user login success/~2023-07-12 12:22:58/~
2 Gowri/~user login success/~2023-07-12 12:23:40/~
3 Gowri/~Select * FROM books/~2023-07-12 12:23:52/~
4 Gowri/~user login success/~2023-07-12 12:51:14/~
5 Gowri/~user login success/~2023-07-12 14:22:09/~
6 Gowri/~user login success/~2023-07-12 14:37:48/~
7 Gowri/~user login success/~2023-07-12 14:39:04/~
8 Gowri/~user login success/~2023-07-12 14:41:09/~
9 Gowri/~user login success/~2023-07-12 14:43:08/~
10 Gowri/~user login success/~2023-07-12 14:44:38/~
11 Gowri/~user login success/~2023-07-12 14:49:00/~
12 Gowri/~CREATE TABLE Movie ( PersonID int, LastName varchar(255), FirstName varchar(255), Address varchar(255), Ci
13 Gowri/~user login success/~2023-07-12 14:57:04/~
14 Gowri/~CREATE TABLE Persons ( PersonID int, LastName varchar, FirstName varchar, Address varchar,City varchar )/
15 Gowri/~user login success/~2023-07-12 14:58:04/~
16 Gowri/~user login success/~2023-07-12 14:58:42/~
17 Gowri/~user login success/~2023-07-12 15:01:54/~
18 Gowri/~CREATE TABLE Pers ( PersonID int, LastName varchar, FirstName varchar, Address varchar, City varchar )/~
19 Gowri/~user login success/~2023-07-12 15:10:15/~
20 Gowri/~CREATE TABLE P ( PersonID int, LastName varchar, FirstName varchar, Address varchar, City varchar )/~202
21 Gowri/~user login success/~2023-07-12 15:15:36/~
22 Gowri/~user login success/~2023-07-12 15:22:35/~
23 Gowri/~CREATE TABLE sons ( PersonID int, LastName varchar )/~2023-07-12 15:24:07/~
24 Gowri/~user login success/~2023-07-12 15:24:56/~
25 Gowri/~CREATE TABLE ns ( PersonID int, LastName varchar)~/2023-07-12 15:25:11/~
26 Gowri/~user login success/~2023-07-12 15:30:10/~
```

Fig 9: logs stored in files with delimiters

Delimiters are used to separate the data stored in a file. It can make reading easier and access the data within the file efficiently.

7. Implementation of queries:

Create query:

To create database, the query used is:

```
CREATE DATABASE database_name;
```

The class uses `getDatabaseName()` which uses `querysection` array that splits the query into an array and returns the 3rd String in array.

It uses `createDatabase()` function which creates a folder with the database name.

```
Enter answer: blue
Login successful! Welcome, Gowri!
Please press 1 to enter query or press 0 to exit:
1
Please enter the query:
CREATE DATABASE goDataBase
query:CREATE DATABASE goDataBase
Created Database:true
```

Fig 10: CREATE DATABASE output

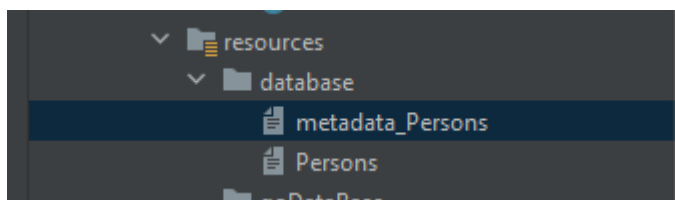


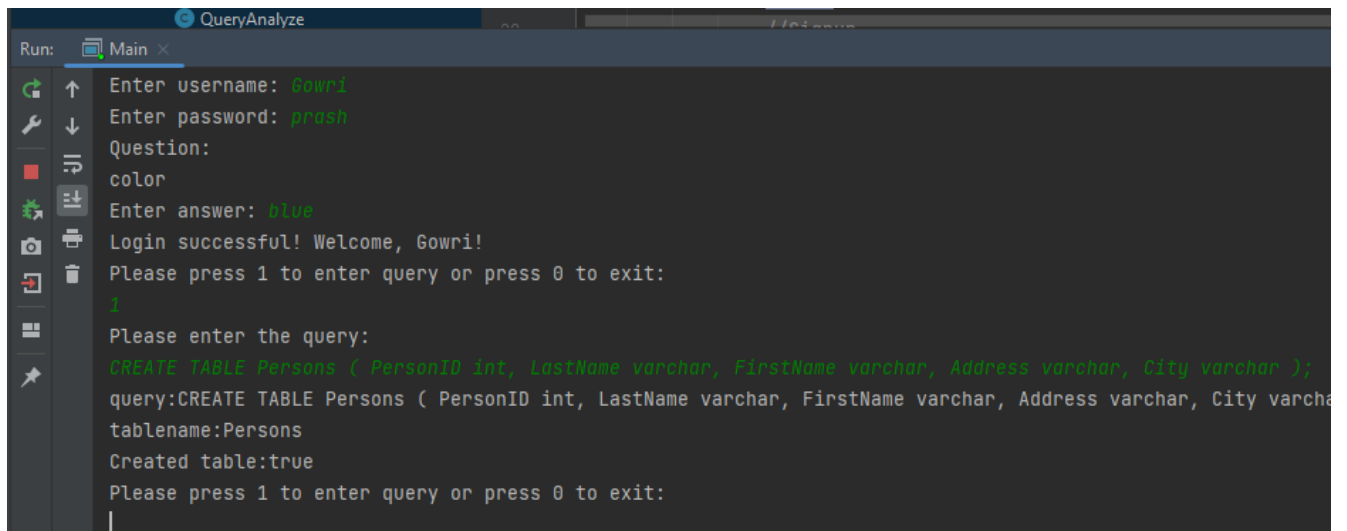
Fig 11: database folder

The database is created and tables can be created with following method:

```
CREATE TABLE Persons ( PersonID int, LastName varchar, FirstName varchar, Address varchar, City varchar );
```

The class uses `getTableName()` which uses `querysection` array that splits the query into an array and returns the 3rd String.

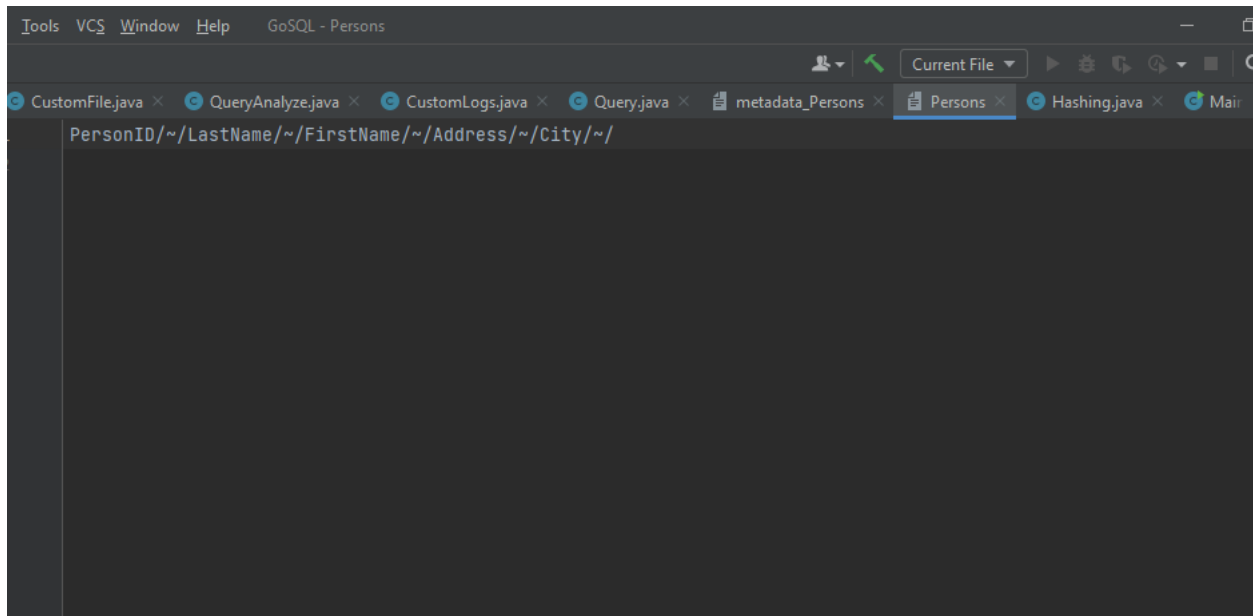
It uses `createTable` function which creates a file with the table name.



```
Run: Main x
Enter username: Gowri
Enter password: prash
Question:
color
Enter answer: blue
Login successful! Welcome, Gowri!
Please press 1 to enter query or press 0 to exit:
1
Please enter the query:
CREATE TABLE Persons ( PersonID int, LastName varchar, FirstName varchar, Address varchar, City varchar );
query:CREATE TABLE Persons ( PersonID int, LastName varchar, FirstName varchar, Address varchar, City varchar );
tablename:Persons
Created table:true
Please press 1 to enter query or press 0 to exit:
|
```

Fig 12: Create table query output

The table file is created with the delimiters



```
Tools  VCS  Window  Help  GoSQL - Persons
Current File
CustomFile.java  QueryAnalyze.java  CustomLogs.java  Query.java  metadata_Persons  Persons  Hashing.java  Main
PersonID/~LastName/~FirstName/~Address/~City/~/
```

Fig 13: table created and stored

Similarly, metadata file is created.

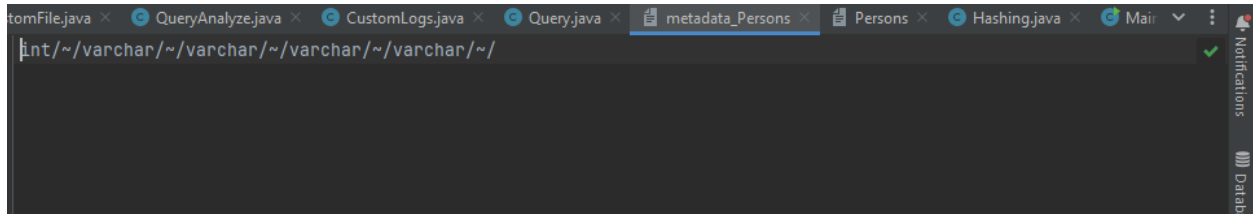


Fig 14: meta-data for table created and stored

Select query:

SELECT * FROM tableName;

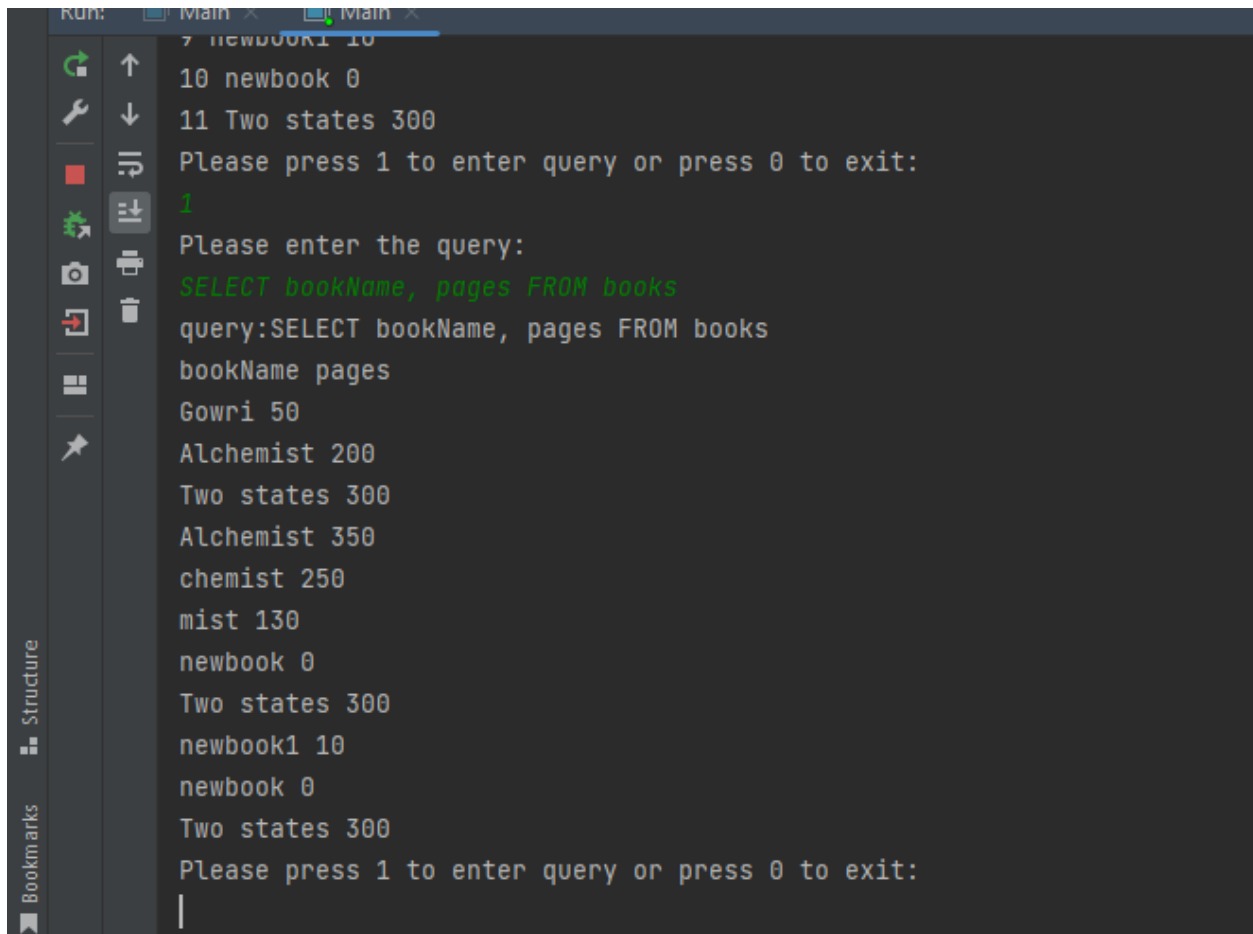
The class uses getTableNameSelect() which uses querysection array that splits the query into an array and returns the 4th String in array.

It uses readFileContents function which reads content of file with the table name and returns a 2D Array that is printed in console

```
Enter answer: blue
Login successful! Welcome, Gowri!
Please press 1 to enter query or press 0 to exit:
1
Please enter the query:
SELECT * FROM books;
query:SELECT * FROM books
bookId bookName pages
1 Gowri 50
2 Alchemist 200
3 Two states 300
4 Alchemist 350
5 chemist 250
6 mist 130
7 newbook 0
8 Two states 300
9 newbook1 10
10 newbook 0
11 Two states 300
Please press 1 to enter query or press 0 to exit:
```

Fig 15: Select query output

SELECT bookName, pages FROM books;



```
Run: Main x Main x
7 newbook1 10
10 newbook 0
11 Two states 300
Please press 1 to enter query or press 0 to exit:
1
Please enter the query:
SELECT bookName, pages FROM books
query:SELECT bookName, pages FROM books
bookName pages
Gowri 50
Alchemist 200
Two states 300
Alchemist 350
chemist 250
mist 130
newbook 0
Two states 300
newbook1 10
newbook 0
Two states 300
Please press 1 to enter query or press 0 to exit:
|
```

Fig 16: Select column query output

Insert query:

INSERT INTO books VALUES (12,newbook,100)

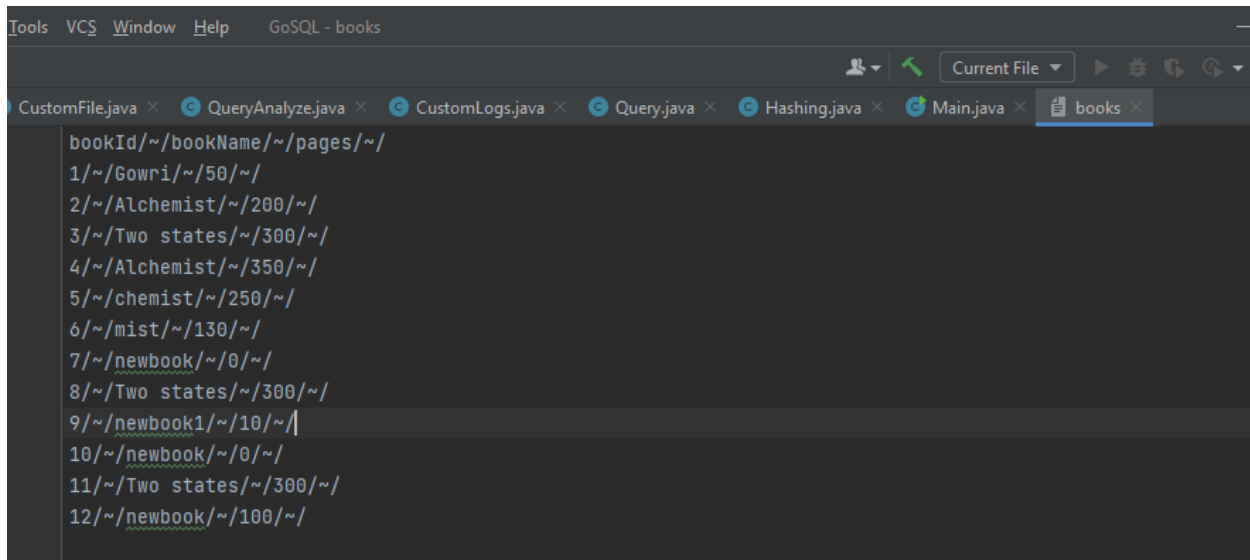
The class uses getTableName() which uses querysection array that splits the query into an array and returns the 3th String in array.

It uses getValues() function while splits the values between the round brackets and returns a string array.

It uses writeFileValues function which writes data below the columns

```
Login successful! Welcome, user1!
Please press 1 to enter query or press 0 to exit:
1
Please enter the query:
INSERT INTO books VALUES (12,newbook,100)
query:INSERT INTO books VALUES (12,newbook,100)
Data written to table successfully.
Please press 1 to enter query or press 0 to exit:
1
```

Fig 17: INSERT query output



```
bookId~/bookName~/pages~/
1~/Gowri~/50~/
2~/Alchemist~/200~/
3~/Two states~/300~/
4~/Alchemist~/350~/
5~/chemist~/250~/
6~/mist~/130~/
7~/newbook~/0~/
8~/Two states~/300~/
9~/newbook1~/10~/
10~/newbook~/0~/
11~/Two states~/300~/
12~/newbook~/100~/
```

Fig 17: Data stored in file after insert query

Update query:

UPDATE books SET bookName=Two states, pages=300 WHERE bookId = 7;

The class uses getTableName() which uses querysection array that splits the query into an array and returns the 2th String in array.

It uses getUpdates() function which splits column name and column value.

It uses getWhereUpdate function which splits the where condition part.

The values are then updated for the given conditions.

```
Enter answer: blue
Login successful! Welcome, Gowri!
Please press 1 to enter query or press 0 to exit:
1
Please enter the query:
UPDATE books SET bookName=Ikigai, pages=300 WHERE bookId = 7
query:UPDATE books SET bookName=Ikigai, pages=300 WHERE bookId = 7
Data written to table successfully.
Update query executed successfully.
Please press 1 to enter query or press 0 to exit:
```

Fig 17: Update query output

```
CustomFile.java x QueryAnalyze.java x CustomL
bookId/~bookName/~pages/~
1/~Gowri/~50~/
2/~Alchemist/~200~/
3/~Two states/~300~/
4/~Alchemist/~350~/
5/~chemist/~250~/
6/~mist/~130~/
7/~Ikigai/~300~/
8/~Two states/~300~/
9/~newbook1/~10~/
10/~newbook/~0~/
11/~Two states/~300~/
12/~newbook/~100~/
```

Fig 18: Data stored in file after update query

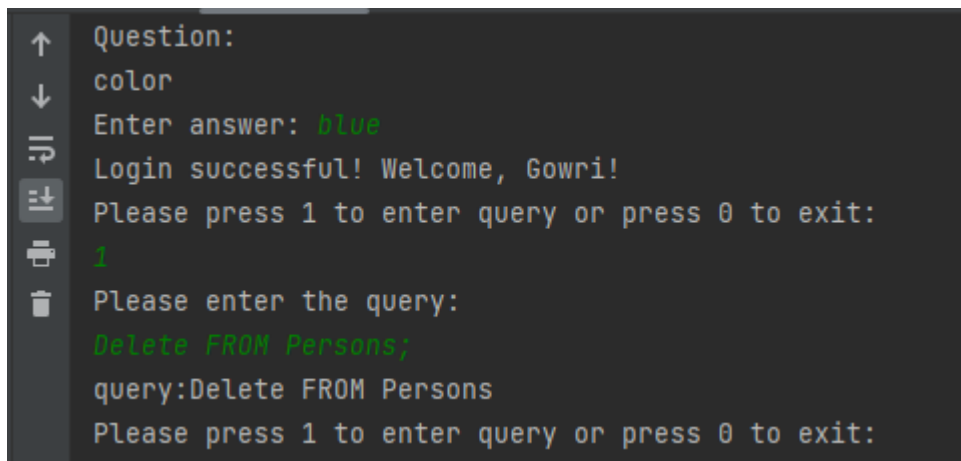
Delete Query:

DELETE FROM books;

The query will delete the contents for the file.

The class uses `getTableName()` which uses `querysection` array that splits the query into an array and returns the 3th String in array.

It uses `deleteFileContents()` which will delete the data in the table.



```
↑ Question:
↓ color
Enter answer: blue
Login successful! Welcome, Gowri!
Please press 1 to enter query or press 0 to exit:
1
Please enter the query:
Delete FROM Persons;
query:Delete FROM Persons
Please press 1 to enter query or press 0 to exit:
```

Fig 19: Data stored in file after delete query

References:

- [1] " SOLID: The First 5 Principles of Object Oriented Design," *Digital Ocean*, [Online]. Available: <https://www.digitalocean.com/community/conceptual-articles/s-o-l-i-d-the-first-five-principles-of-object-oriented-design#single-responsibility-principle> [Accessed: June 29,2023]
- [2] " SOLID Principles: Open Closed Principle for better Understanding of extension and modification.," *Knoldus*, [Online]. Available: <https://blog.knoldus.com/solid-open-closed-principle/> [Accessed: June 29,2023]
- [3] "The Liskov Substitution Principle Explained," *reflectoring*, [Online]. Available: <https://reflectoring.io/lsp-explained/> [Accessed: July 02,2023]
- [4] "SOLID: Interface Segregation Principle," *TutorialsTeacher*, [Online]. Available: <https://www.tutorialsteacher.com/csharp/interface-segregation-principle> [Accessed: July 03,2023]
- [5] " Dependency Inversion Principle," *DevIq*, [Online]. Available: <https://deviq.com/principles/dependency-inversion-principle> [Accessed: July 05,2023]
- [6] "Java Hashing using MD5," *howtodoinjava*, [Online]. Available: <https://howtodoinjava.com/java/java-security/how-to-generate-secure-password-hash-md5-sha-pbkdf2-bcrypt-examples/> [Accessed: July 05,2023]
- [7] "How to split a String in Java?," *codeahoy*, [Online].Available: <https://www.geeksforgeeks.org/split-string-java-examples/>
- [8] "How to load text data into a table using Java" *stackoverflow*, [Online].Available: <https://stackoverflow.com/questions/31043165/how-to-load-text-data-into-a-table-using-java>