

CSCI 5408

Assignment 1

Sarthak Patel

sr555161@dal.ca

Problem – 1 Report

No.	Entity	Attributes & Reason
1	Dalhousie University	<p>Attributes: <u>id</u>, address, Dean, contact.</p> <p>Reason: Dalhousie university is the central entity of the whole ERD Diagram. All the other entities are taken from the Dalhousie university website (https://www.dal.ca/). University has all these attributes because there is also a Dalhousie branch at other locations apart from Halifax, like in Truro.</p>
2	Campus	<p>Attributes: <u>id</u>, name, size, location, facilities.</p> <p>Reason: Dalhousie university has many campuses in Halifax like, Sexton Campus, Studly Campus etc. And they provide multiple facilities.</p>
3	Faculty & Staff	<p>Attributes: <u>id</u>, name, contact, expertise area.</p> <p>Reason: Dalhousie University has staff entity which stores all the essential details of the faculty or staff.</p>
4	Admissions	<p>Attributes: <u>id</u>, name, requirements, acceptance rate, scholarship</p> <p>Reason: This entity is responsible for storing all the details of the program and the courses offered by the university. Every program has its different requirements, acceptance rate and not all programs provide scholarship.</p>

5	Academic Course	<p>Attributes: <u>id</u>, faculty id, requirements, curriculum.</p> <p>Reason: Dalhousie provides multiple courses like Data management and Warehousing. The course has its prerequisites, assigned professor and structured curriculum to follow. This entity is responsible for storing such information.</p>
6	Alumni Board (<i>weak</i>)	<p>Attributes: <u>id</u>, name, position, qualification.</p> <p>Reason: Dalhousie has alumni board entity which stores all the data of the board members.</p>
7	Research & Innovation	<p>Attributes: <u>id</u>, areas, funding, center.</p> <p>Reason: Research & Innovation entity stores all the information related to research areas, funding and center offered by the university. Dalhousie provides research opportunities.</p>
8	Libraries	<p>Attributes: <u>id</u>, name, address, hours.</p> <p>Reason: Library entity is responsible for storing all the details of the library handled by the Dalhousie university. Every library has name, address and their working hours.</p>
9	Books (<i>weak</i>)	<p>Attributes: <u>id</u>, name, author, version.</p> <p>Reason: Books entity store all the data of books present in library with their name, author and their latest edition.</p>

10	Laptop (<i>weak</i>)	<p>Attributes: <u>id</u>, name, student id, start date, end date.</p> <p>Reason: Dalhousie library provides laptop to the students for the certain period.</p>
11	Online Resources (<i>weak</i>)	<p>Attributes: <u>id</u>, department.</p> <p>Reason: Online resources entity is responsible for storing the details of the online resources accessed from which department.</p>
12	News & Events	<p>Attributes: <u>id</u>, name, social media links, newsletters, event date.</p> <p>Reason: News & Events entity stores all the events and the news related to them along with the social media links.</p>
13	Contact Us	<p>Attributes: phone no., email, physical address, feedback form, campus map.</p> <p>Reason: Contact us entity stores the different ways to contact the university along with providing feedback form.</p>

Relationships

1. Dalhousie University is in one-to-many relationship with Campus.
2. Dalhousie University is in one-to-many relationship with Faculty & Staff.
3. Dalhousie University is in one-to-many relationship with Admissions.
4. Dalhousie University is in one-to-many relationship with Academic Course.
5. Admissions is in one-to-one relationship with Alumni Board.
6. Dalhousie University is in one-to-many relationship with Research & Innovation.
7. Dalhousie University is in one-to-many relationship with Libraries.
8. Libraries is in one-to-many relationship with Books.
9. Libraries is in one-to-many relationship with Laptop.
10. Libraries is in one-to-many relationship with Online Resources.
11. Dalhousie University is in one-to-many relationship with News & Events.
12. Dalhousie University is in one-to-many relationship with Contact Us.

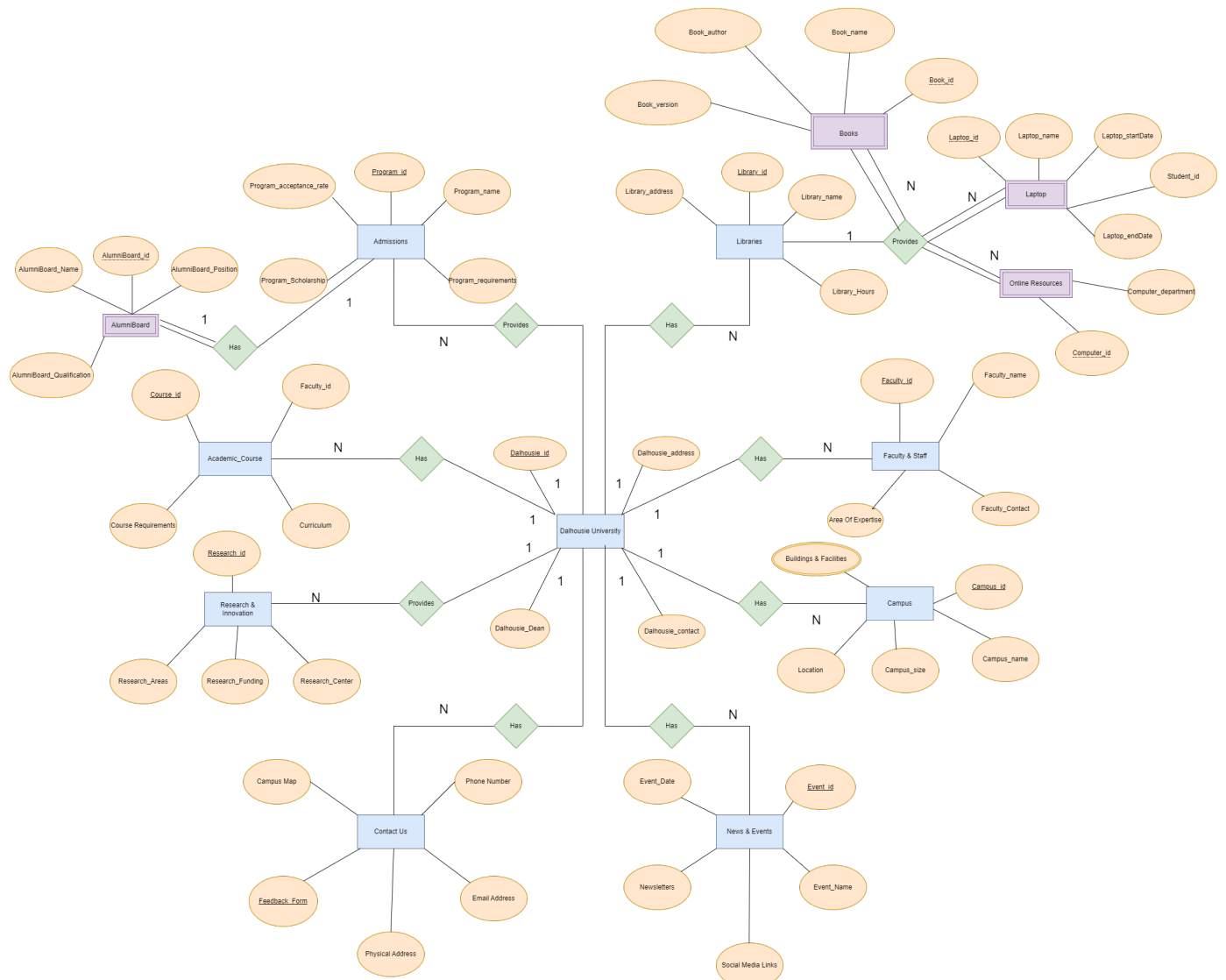


Fig. 1: Conceptual Model for the Dalhousie university.

Note: For Better view of the above conceptual model click on the below attached link.

https://drive.google.com/file/d/1tb3L1SnIErGo5pn4kzt4_kY-tNvnXZmw/view?usp=sharing

Normalization In Dalhousie DBMS

Campus
Campus_Id
Campus_name
Campus_size
Campus_Location
Buildings & Facilities

Facility
Campus_id
Facility

Entity before normalized

1 NF Normalized

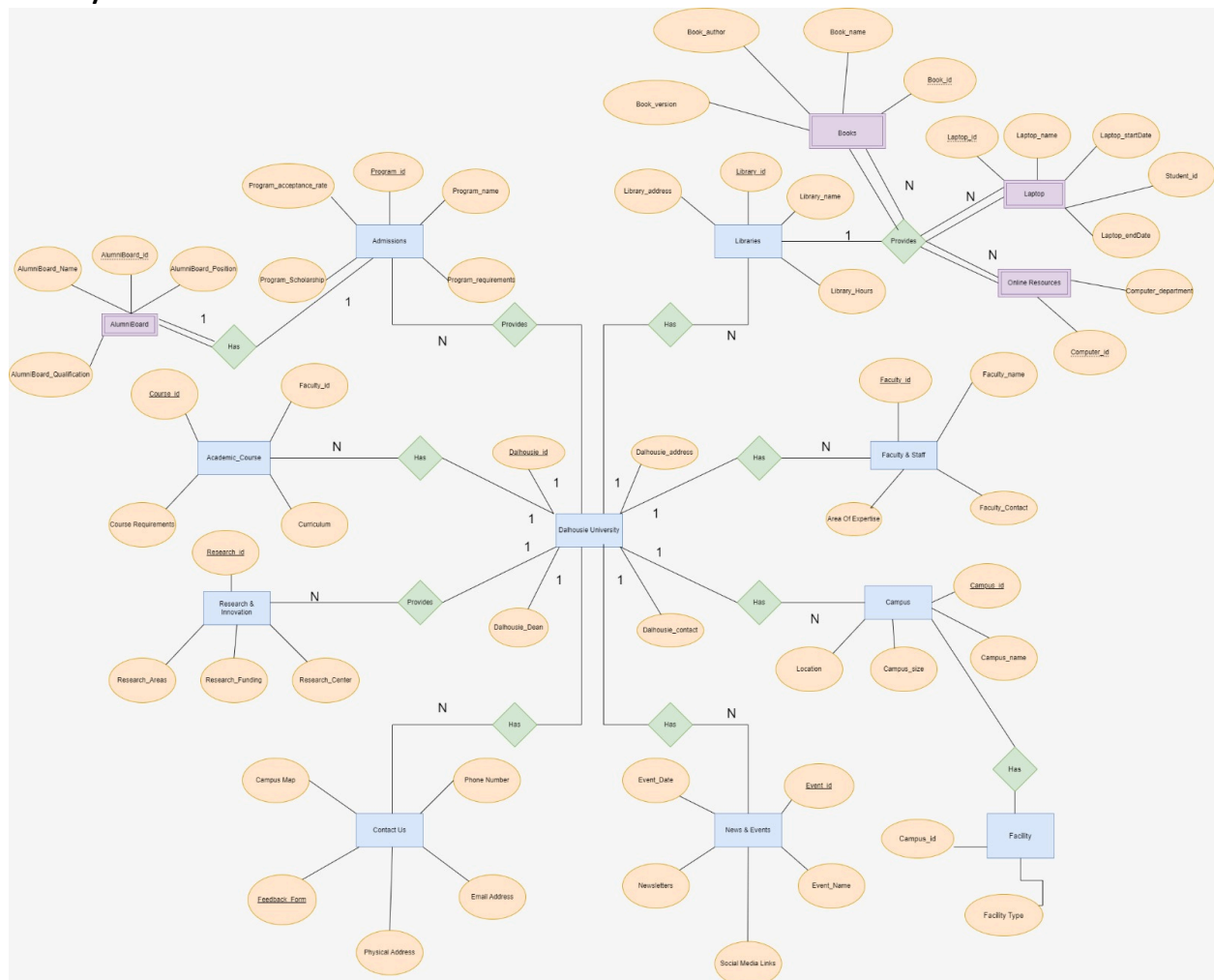


Fig2: Conceptual Model after 1 NF form.

Note: For Better view of the above conceptual model click on the below attached link.

<https://drive.google.com/file/d/1MPTJs0Ypgvp-nHHy2SMsLFU6Bup8NXcM/view?usp=sharing>

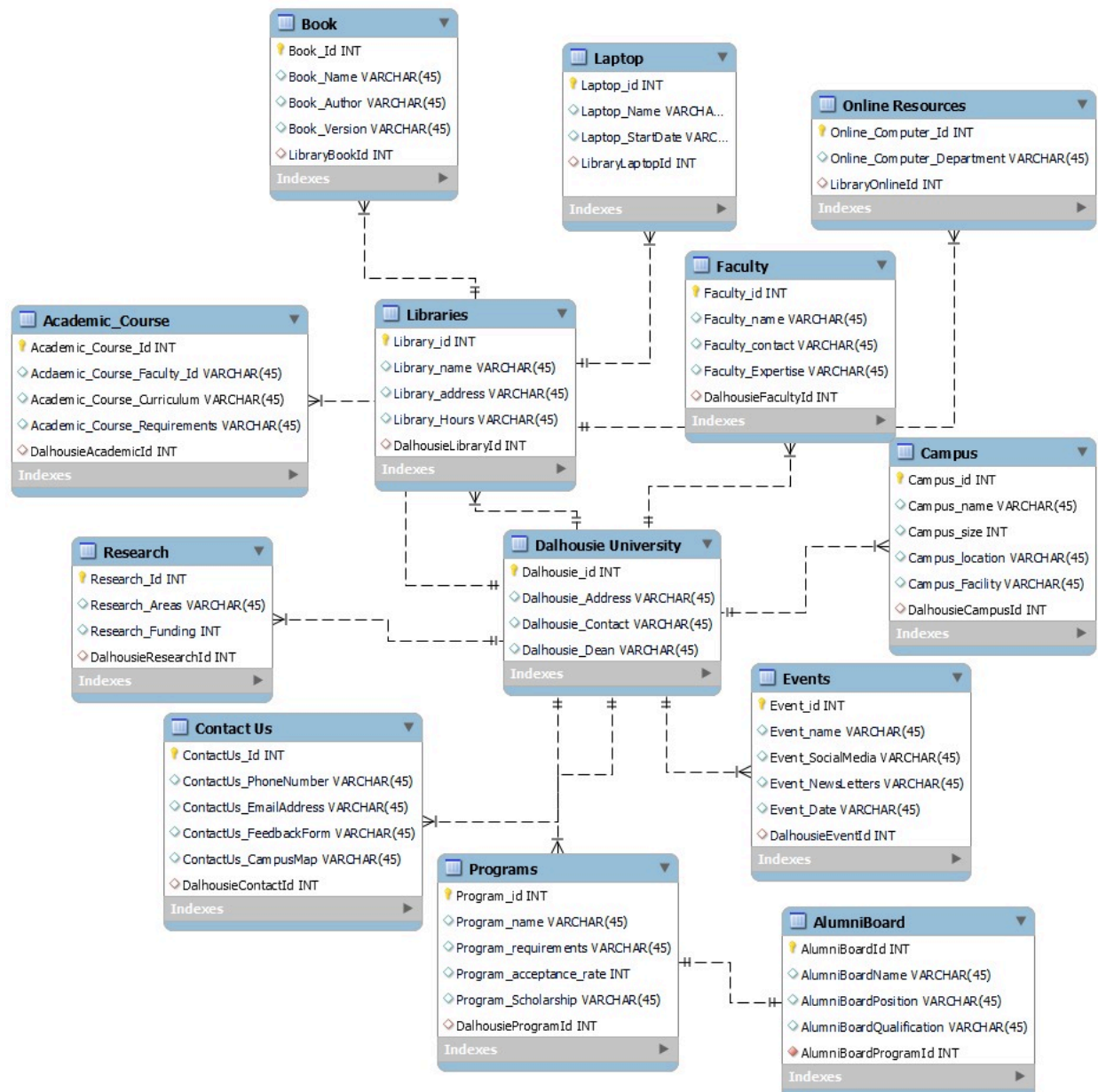


Fig 3: Logical Model for the Dalhousie DBMS.

PROBLEM – 2

Aim

To create a prototype of a light-weight DBMS using java programming language without the use of third-party libraries.

Approach

To make the system persistent like DBMS, it is best to store the data in files with the help of File.io library of java. Application is console based with the multifactor authentication. With the help of regex and split method of the java the query is processed to remove the unnecessary keyword like [, ";] and many more. With the help of that only important keyword like "CREATE", "FROM", table name, table columns and attributes are stored with the help of java data structures. There are chances that the user has input wrong query. So, a validation for query format has been done with the help of the regex.

The program defines a class called Query that has three instance variables: id, path, and DBMSName. The id variable is a string that represents the unique identifier of each record. The path variable stores the location of the data file, while the DBMSName variable stores the name of the database management system.

The Query class has three methods: CREATE, SELECT, and INSERT. The CREATE method takes a command as an argument, which is a string containing the SQL CREATE statement. The method parses the command string to extract the table name and the column names and creates a new file in the specified path with the table name as the file name. The method then writes the column names to the file and prints a success message if the file is created successfully.

The SELECT method takes a command string as an argument, which is a string containing the SQL SELECT statement. The method parses the command string to extract the table name and the column names to be selected. The method then reads the data from the file and stores it in a LinkedHashMap. The method then retrieves the columns to be displayed and prints the data to the console.

The INSERT method takes a command string as an argument, which is a string containing the SQL INSERT statement. The method parses the command string to extract the table name, column names, and values to be inserted. The method reads the column names from the file and retrieves the last ID number to generate a new ID for the new record. The method then writes the new record to the file and prints a success message if the record is inserted successfully.

Extra Features

- Multi-factor authentication.
- The logged in user can only access the data bases created by him. For example: If user A has created a database (folder), “ADBMS” then the user B even if authenticated cannot access the tables of database ADBMS. This was done to make the system more robust.
- By default, auto increment column by the name of the “index” has been implemented to imitate the auto-increment feature of MySQL Workbench.
- User Don’t need to create database by writing query. It gets automatically created with its ID.
- Like a database for an insert query if a column is not mentioned while inserting it is automatically marked as null. This feature has been implemented in the java program.

Limitations

- Data type in queries are not included to reduce the complexity of the project without compromising the functionality of the system.
- Single Valued Entries are accepted. For example: If I must insert name, Sarthak in the table. Then I can't insert "Sarthak Patel". I can only insert "Sarthak" or "Patel". This has been done to reduce the complexity of the code and implement the functionality in minimalistic way.
- WHERE clause for SELECT query has not been implemented but it has been implemented for UPDATE and DELETE queries.
- One Query can be fired at a time. Multiple insert or other queries can't be fired.

Expected Query Inputs

1) CREATE Query

- CREATE TABLE table_name (column1, column2,);
 - Example: CREATE TABLE MOVIES (id, movie, actor, ratings);

2) INSERT Query

- INSERT INTO table_name (column1, column2,...) VALUES (value1, value2,);
 - Example: INSERT INTO MOVIES (id, movie, actor, ratings) VALUES ('crjor5', 'Creed', 'Jordan', '5');

3) SELECT Query

- SELECT * FROM table_name;
 - Example: SELECT * FROM MOVIES;
- SELECT column1, column2,.....FROM table_name;
 - Example: SELECT index, movie, ratings FROM MOVIES;

4) UPDATE Query

- UPDATE table_name SET column1 = value1, column2 = value2, WHERE condition;
- UPDATE MOVIES SET actor = 'Timothée', movie = 'Dune' WHERE id = 2;

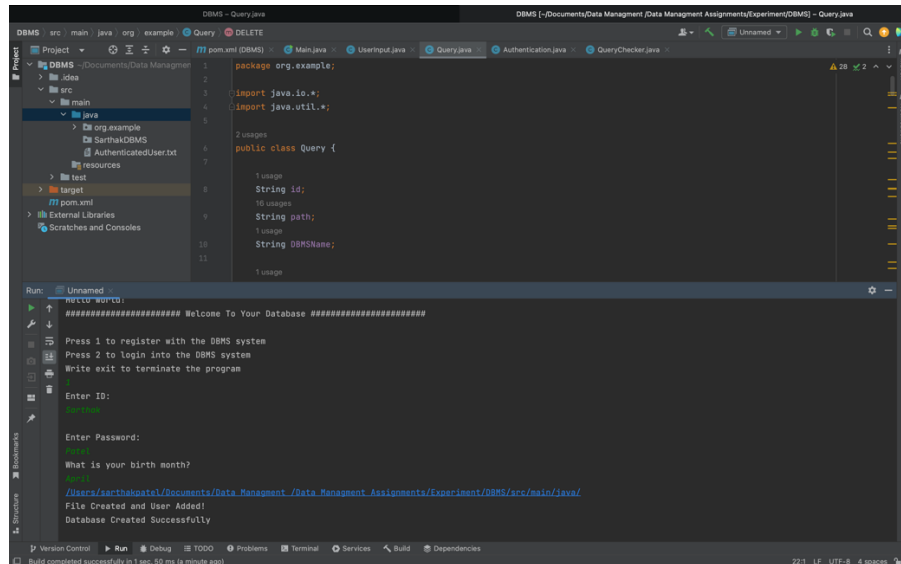
5) DELETE Query

- DELETE FROM table_name WHERE condition;
- DELETE FROM MOVIES WHERE actor = Jordan;

NOTE: When Create Table query is fired. It is fired successfully but the created table won't show in IntelliJ application until the destination folder where this table is created is not open. I don't know if the problem is in my system or not. Once done the above, everything works perfectly fine.

TEST CASES

1) Adding User



```
package org.example;

import java.io.*;
import java.util.*;

public class Query {
    String id;
    String path;
    String DBMSName;
}
```

```
Run: Unnamed
Hello world!
##### Welcome To Your Database #####

Press 1 to register with the DBMS system
Press 2 to login into the DBMS system
Write exit to terminate the program

Enter ID:
Sarthak

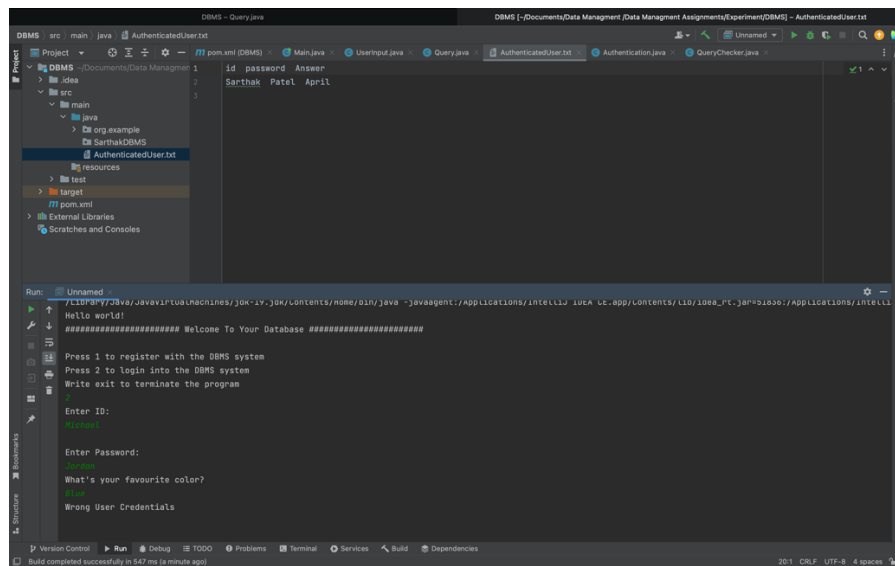
Enter Password:

What is your birth month?
/

/Users/sarthakpatel/Documents/Data Management /Data Management Assignments/Experiment/DBMS/src/main/java/
File Created and User Added!
Database Created Successfully
```

Fig 4: Adding user to the system.

2) Authenticating User



```
id password Answer
Sarthak Patel April
```

```
Run: Unnamed
Hello world!
##### Welcome To Your Database #####

Press 1 to register with the DBMS system
Press 2 to login into the DBMS system
Write exit to terminate the program

Enter ID:
Sarthak

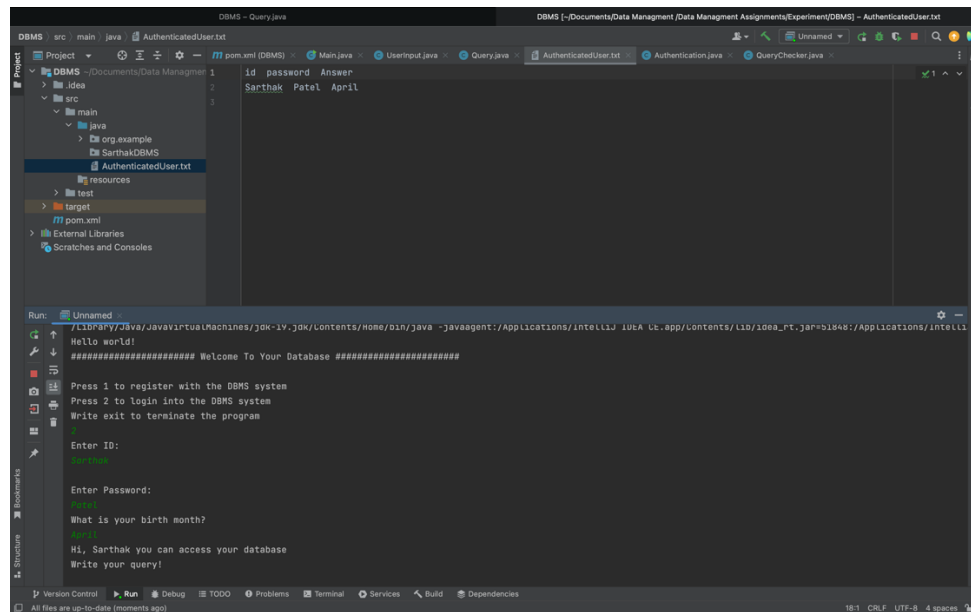
Enter Password:

What's your favourite color?
Red

Wrong User Credentials
```

Fig 5: Checking user entry in database.

3) Login User



```
DBMS - Query.java
DBMS [-Documents/Data Management/Data Management Assignments/Experiment/DBMS] - AuthenticatedUser.txt

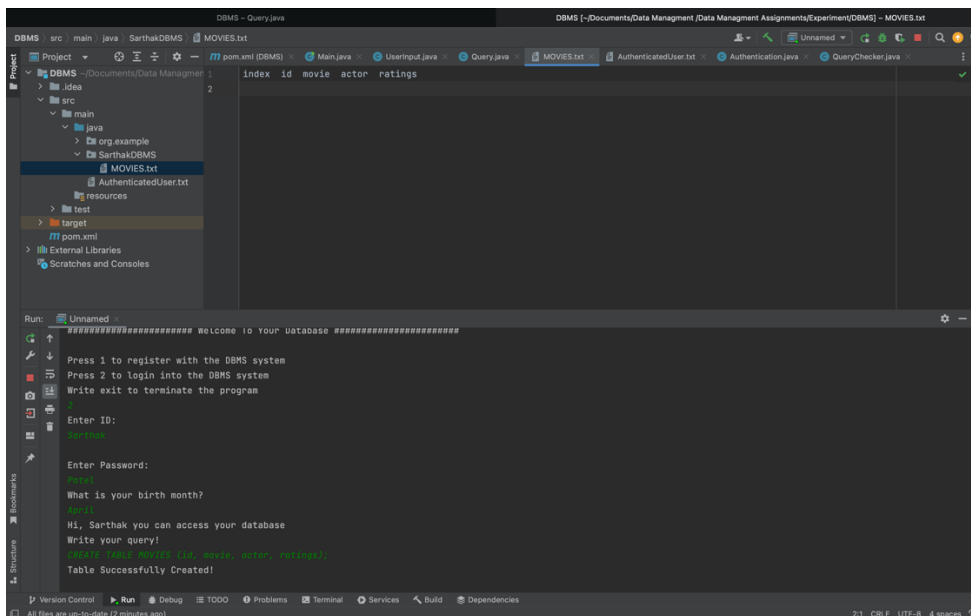
Project | DBMS | pom.xml (DBMS) | Main.java | UserInput.java | Query.java | AuthenticatedUser.txt | Authentication.java | QueryChecker.java
src | main | java | id | password | Answer
1 | Sarthak | Patel | April
2 |
3 |

Run: Unnamed -
/.../lib/Java/javavirtualmachines/jdk-17-jdk/Contents/Home/bin/java -javaagent:/Applications/IntelliJ...
Hello world!
##### Welcome To Your Database #####

Press 1 to register with the DBMS system
Press 2 to login into the DBMS system
Write exit to terminate the program
Enter ID:
Sarthak
Enter Password:
123456
What is your birth month?
April
Hi, Sarthak you can access your database
Write your query!
```

Fig 6: Logging in user to the database.

4) Create Table



```
DBMS - Query.java
DBMS [-Documents/Data Management/Data Management Assignments/Experiment/DBMS] - MOVIES.txt

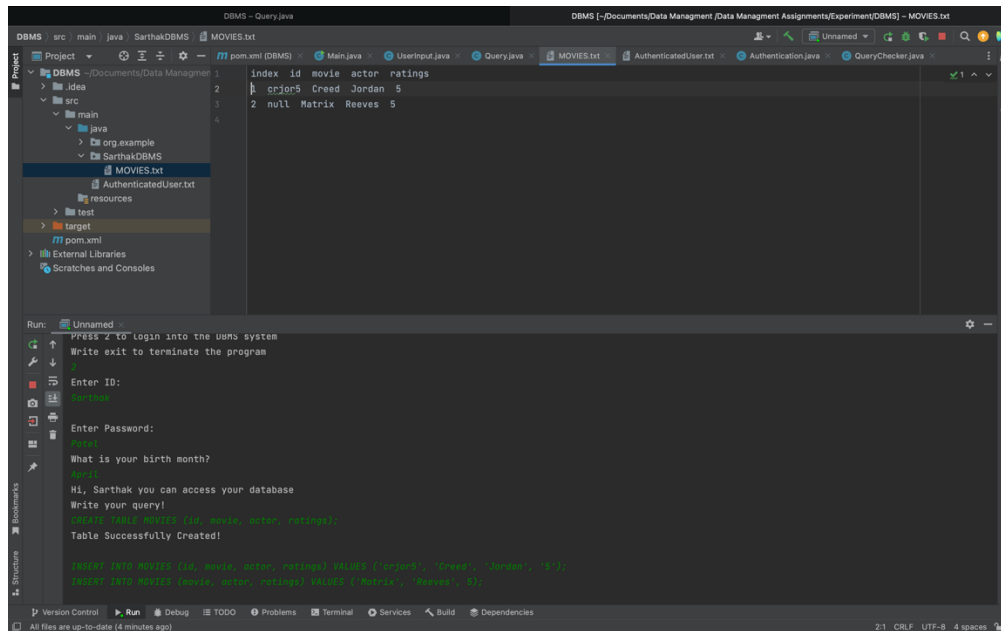
Project | DBMS | pom.xml (DBMS) | Main.java | UserInput.java | Query.java | AuthenticatedUser.txt | Authentication.java | QueryChecker.java
src | main | java | SarthakDBMS | MOVIES.txt
1 | index | id | movie | actor | ratings
2 |
3 |

Run: Unnamed -
##### Welcome to Your Database #####

Press 1 to register with the DBMS system
Press 2 to login into the DBMS system
Write exit to terminate the program
Enter ID:
Sarthak
Enter Password:
123456
What is your birth month?
April
Hi, Sarthak you can access your database
Write your query!
CREATE TABLE MOVIE (
index INT,
id VARCHAR(20),
movie VARCHAR(50),
actor VARCHAR(50),
ratings INT);
Table Successfully Created!
```

Fig 7: Creating the table named MOVIE

5) Inserting Value in Table



```
DBMS - Query.java
DBMS [-/Documents/Data Management/Data Management Assignments/Experiment/DBMS] - MOVIES.txt

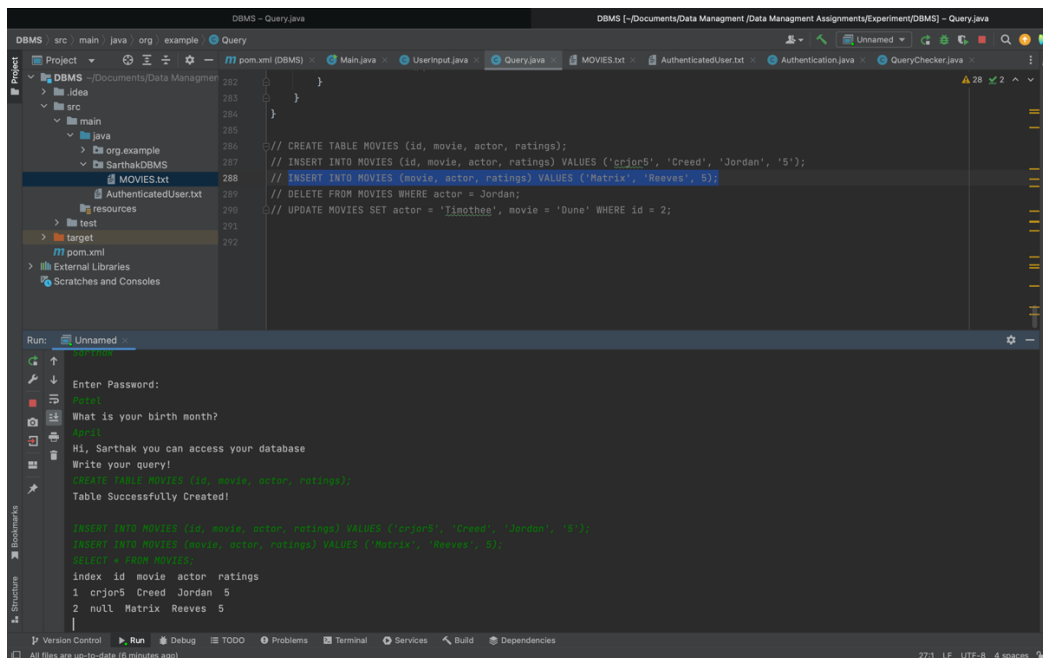
Project | pom.xml (DBMS) | Main.java | UserInput.java | Query.java | MOVIES.txt | AuthenticatedUser.txt | Authentication.java | QueryChecker.java

DBMS - Query.java
1 index id movie actor ratings
2 1 crjor5 Creed Jordan 5
3 2 null Matrix Reeves 5
4

Run: Unnamed
Press 2 to login into the ums system
Write exit to terminate the program
Enter ID:
1
Enter Password:
123456
What is your birth month?
12
Hi, Sarthak you can access your database
Write your query!
CREATE TABLE MOVIES (id, movie, actor, ratings);
Table Successfully Created!
INSERT INTO MOVIES (id, movie, actor, ratings) VALUES ('crjor5', 'Creed', 'Jordan', 5);
INSERT INTO MOVIES (id, movie, actor, ratings) VALUES (2, 'Matrix', 'Reeves', 5);
```

Fig 8: Inserting the entry in MOVIES table.

6) Selecting Value in Table



```
DBMS - Query.java
DBMS [-/Documents/Data Management/Data Management Assignments/Experiment/DBMS] - Query.java

Project | pom.xml (DBMS) | Main.java | UserInput.java | Query.java | MOVIES.txt | AuthenticatedUser.txt | Authentication.java | QueryChecker.java

DBMS - Query.java
282 }
283 }
284 }
285 }
286 // CREATE TABLE MOVIES (id, movie, actor, ratings);
287 // INSERT INTO MOVIES (id, movie, actor, ratings) VALUES ('crjor5', 'Creed', 'Jordan', 5);
288 // INSERT INTO MOVIES (movie, actor, ratings) VALUES ('Matrix', 'Reeves', 5);
289 // DELETE FROM MOVIES WHERE actor = Jordan;
290 // UPDATE MOVIES SET actor = 'Timothee', movie = 'Dune' WHERE id = 2;
291
Run: Unnamed
Enter Password:
123456
What is your birth month?
12
Hi, Sarthak you can access your database
Write your query!
CREATE TABLE MOVIES (id, movie, actor, ratings);
Table Successfully Created!
INSERT INTO MOVIES (id, movie, actor, ratings) VALUES ('crjor5', 'Creed', 'Jordan', 5);
INSERT INTO MOVIES (movie, actor, ratings) VALUES ('Matrix', 'Reeves', 5);
SELECT * FROM MOVIES;
index id movie actor ratings
1 crjor5 Creed Jordan 5
2 null Matrix Reeves 5
```

Fig 9: Selecting all the values in table.

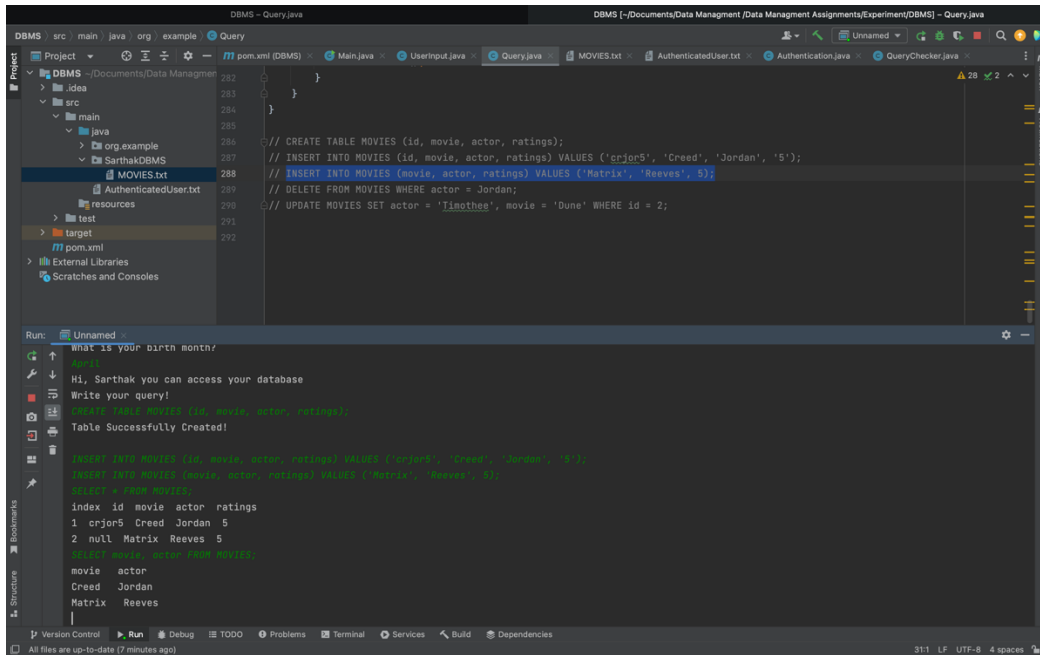


Fig 10: Selecting with columns value in table MOVIE.

7) Updating Value in Table

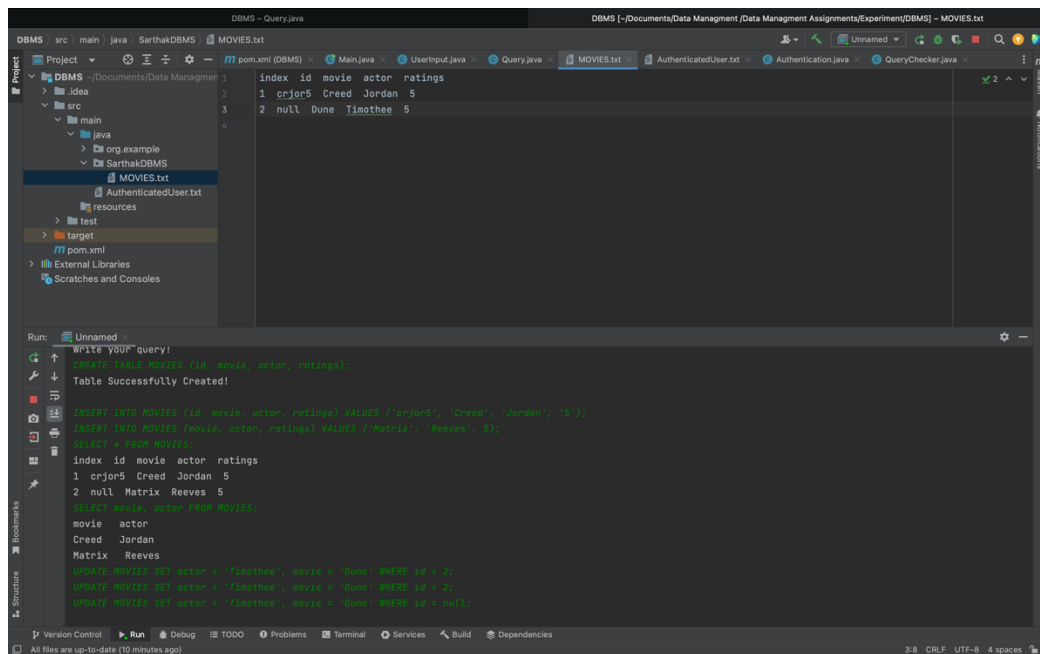


Fig 11: Updating the value in table MOVIE.

8) Delete Value in Table

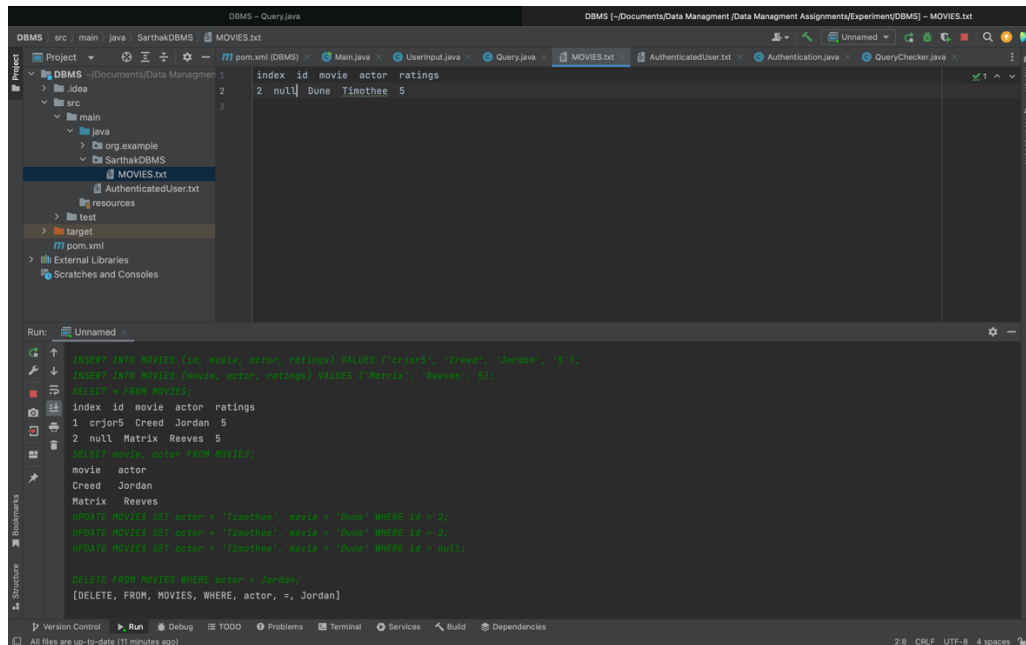


Fig 12: Deleting the entry in table MOVIE.

References

- [1] *Dalhousie University*. [Online]. Available: <https://www.dal.ca/>. [Accessed: 02-Mar-2023].
- [2] “Draw.io - free flowchart maker and diagrams online,” *Flowchart Maker & Online Diagram Software*. [Online]. Available: <https://app.diagrams.net/>. [Accessed: 02-Mar-2023].

Tools Used

- 1) MySQL Workbench
- 2) Draw.io