**DB connection Document**

DB Connection steps:

**Import JDBC packages:**

Import the required package,

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.Statement;

**A screenshot of a cell phone

Description automatically generated**

**Load and register the JDBC driver:** Since JDBC 4 explicitly registering the driver is optional, we just need to vendor’s Jar in the class path, and then JDBC driver can detect and load the driver automatically.

A screenshot of a social media post

Description automatically generated

**Open a connection to the database:** In our application we have created A separate function inside datasource.java class in order to call open() wherever necessary.

A screenshot of a cell phone

Description automatically generated

**Create String constants to hold connection and query related string constant:**

**A screenshot of a social media post

Description automatically generated**

**Create a statement object to perform a query:**We have created different functions for different operations like select, update, insert, delete so that they can be calledwithin respective modules of our application.

**queryProducts()**

**A screenshot of a social media post

Description automatically generated**

This function will query the database using select statement to display all the products in the table.

**UpdateProducts():**

**A screenshot of a social media post

Description automatically generated**

This query will update the products in the table using update query using the inputs given by user.

insertProducts()

A screenshot of a social media post

Description automatically generated

Used to insert the products into table using the user inputs.

**deleteProducts()**

A screenshot of a cell phone

Description automatically generated

Delete the products based on the product ID given by the user.

**Execute the statement object and return a query result set:**

We are holding the result set of select query inside an array list using that in the appropriate places.

**try**(Statement statement = *conn*.createStatement();

ResultSet results = statement.executeQuery(sb.toString())) {

**while**(results.next()) {

ProdDetail product = **new** ProdDetail();

product.setProductID(results.getInt(***COLUMN\_PRODUCT\_ID***));

product.setProductInfo(results.getString(***COLUMN\_PRODUCT\_INFO***));

product.setManufacturer(results.getString(***COLUMN\_PRODUCT\_MANU***));

product.setCost(results.getFloat(***COLUMN\_PRODUCT\_COST***));

*products*.add(product);

}

**return** *products*;

} **catch**(SQLException e) {

System.***out***.println("Query failed: " + e.getMessage());

**return** **null**;

}

**Close the connection:**

**public** **void** close() {

**try** {

**if**(*conn* != **null**) {

*conn*.close();

}

} **catch**(SQLException e) {

System.***out***.println("Couldn't close connection: " + e.getMessage());

}

}

We are closing the connection here using the above close().