BINUS UNIVERSITY
BINUS INTERNATIONAL

Assignment Cover Letter

(Individual Work)

Student Information:		**Surname**	**Given Names**	**Student ID Number**
	1.	**Vanessa**	**Vicky**	**2201807791**

Course Code	**: COMP6510**	**Course Name**	**: Programming Language**
Class	**: L2BC**	**Name of Lecturer(s)**	: Minaldi Loeis
Major	**: CS**		
Title of Assignment (if any)	: Library Storage System		
Type of Assignment	**: Final Project**		
Submission Pattern			
Due Date	**: 01 - 07 - 2019**	**Submission Date**	**: 01 – 07 -2019**

The assignment should meet the below requirements.

1. Assignment (hard copy) is required to be submitted on clean paper, and (soft copy) as per lecturer's instructions.
2. Soft copy assignment also requires the signed (hardcopy) submission of this form, which automatically validates the softcopy submission.
3. The above information is complete and legible.
4. Compiled pages are firmly stapled.
5. Assignment has been copied (soft copy and hard copy) for each student ahead of the submission.

Plagiarism/Cheating

BiNus International seriously regards all forms of plagiarism, cheating and collusion as academic offenses which may result in severe penalties, including loss/drop of marks, course/class discontinuity and other possible penalties executed by the university. Please refer to the related course syllabus for further information.

Declaration of Originality

By signing this assignment, I understand, accept and consent to BiNus International terms and policy on plagiarism. Herewith I declare that the work contained in this assignment is my own work and has not been submitted for the use of assessment in another course or class, except where this has been notified and accepted in advance.

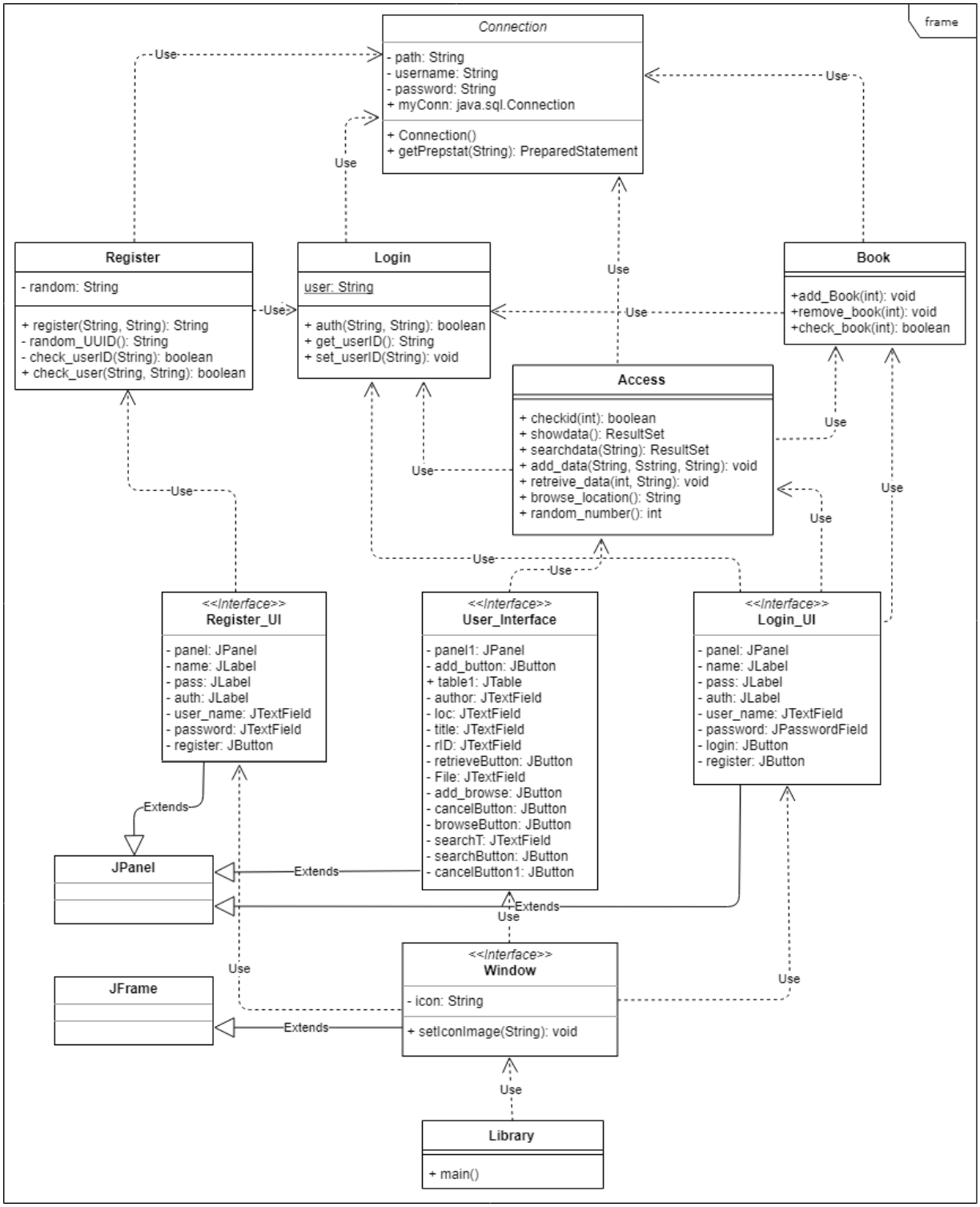
Signature of Student: (Name of Student)
Vicky Vanessa

Table of Contents

I. Description

This project is about a Library Storage System, as the name imply, it is a storage system which store a small sized file like a pdf file in the program, and it can be inserted or retrieved like a library would. This system use database, and graphical user interface. The database is used as the storage system to store the data and information of the file, and GUI is used to make it easier for the user to view and use the program.

II. Solution Design



III. Discussion

Implementation:

- java.sql – to help process the data from and to the database
- java.swing – to make the GUI
- java.awt – to help the layout and the event happening on the GUI
- java.util.UUID – to generate random unique ID for each of the user
- java.util.Random – to generate the random unique ID for the book
- java.io – to retrieve the blob file in the database table and download it to the selected file location

How it works:

- The registering part of the program is for new user, they enter their username and password, and the program will generate a new random ID for every new user that will be stored in the database table.
- When the user login, the program will check the inputted username and password whether it is in the database table or not. If yes, the program will direct the user to their library and load the saved/stored data, else, it will send a warning to them.
- The inserted information considered as book in the program will store a blob file from the data selected. While the data is processed, it will generate random number id for the book to use as identification of the book as book(s) can have the same title, but they will have different identification.
- One of the database table specifically used to store the book(s) under the logged user by their unique ID and the book ID. The program will check which book(s) are connected to the logged user using their ID and will show the user which book(s) belong to them in the program.
- It will delete/retrieve book based on the book ID and check whether the logged user own the specified book in the database or not. Then, it will write the file to the selected file location and specified file name and extension.
- The search function is to search the book(s) by title and update the table with the resulted data every time the user clicked search or cancel.

Code explanation:

- Connection Class

This class is specifically made to connect the java code with the database and to make easier access of it in the other classes.

```
public Connection() {
    try {
        myConn = DriverManager.getConnection(path, username, password);
        myStat = myConn.createStatement();

    } catch (SQLException e) {
        e.printStackTrace();
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

The path is the name of the database in the localhost and the username and password are the one used to open and access the database.

```java
public PreparedStatement getPrepstat(String query) {
    try {
        PreparedStatement prepStat = myConn.prepareStatement(query);
        return prepStat;
    } catch (SQLException e) {
        e.printStackTrace();
    }
    return null;
}
```

This part of the code is for the other class have easier access to use the Prepared Statement.

- Login Class

This class handles the user authentication of whether their account already registered or not.

```java
public boolean auth(String name, String password){
    try {
        PreparedStatement prepStat = connect.getPrepstat( query: "SELECT user_id, user_name, pass FROM login " +
                "WHERE user_name = ? AND pass = ?");
        prepStat.setString( parameterIndex: 1, name);
        prepStat.setString( parameterIndex: 2, password);
        ResultSet myRs = prepStat.executeQuery();
        if(myRs.next()){
            user = myRs.getString( columnLabel: "user_id");
            System.out.println("User " + myRs.getString( columnLabel: "user_name") + " Successfully Logged In!");
            return true;
        }
        else {
            System.out.println("Login Declined!");
            return false;
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }
    return false;
}
```

It will check the username and password inputted by the user and set it to the query. myRs.next() is to see whether there is someone with that data or not. If there is a next, it will set the user string to the user ID and it will return true to get access in the login panel in the code.

```java
public String get_userID() { return user; }
```

Will be used so the other class can get the user ID from this class.

```java
private static String user;
```

This string is set to be static so that even though this class is instantiated by many classes, this string will not change and can be used. If it is not set to be static, by the many instance of this class, it will be set to null and therefore cannot be used.

```java
public void set_userID(String userID) { user = userID; }
```

This method is later on used in the register later to set the user ID to the generated ID by the register class.

- Register Class

This class it used for registering purposes for the new user.

```java
public boolean check_user(String user, String password){
    try {
        PreparedStatement prepStat = connect.getPrepstat( query: "SELECT user_name, pass FROM login " +
                "WHERE user_name = ? AND pass = ?");
        prepStat.setString( parameterIndex: 1, user);
        prepStat.setString( parameterIndex: 2, password);

        ResultSet myRs = prepStat.executeQuery();
        if(myRs.next()){
            return true;
        }
        else{
            return false;
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }
    return false;
}
```

This method is used to check whether there is a user who have taken both the username and password, if there is no one who have taken both the user and the password, it will validate the user to register.

```java
public void register(String user_name, String password){
    try {
        PreparedStatement prepStat = connect.getPrepstat( query: "INSERT INTO login " +
                "VALUES(?,?,?)");
        prepStat.setString( parameterIndex: 1, random);
        prepStat.setString( parameterIndex: 2, user_name);
        prepStat.setString( parameterIndex: 3, password);
        int i = prepStat.executeUpdate();
        if(i > 0){
            login.set_userID(random);
            System.out.println( "Successfully Registered!");
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }
}
```

This method is used to input the validated register into the database. After it successfully recorded in the database, it will set the generated user ID to the login class.

```java
private String random_UUID(){
    UUID uuid = UUID.randomUUID();
    String random = uuid.toString();
    while(true){
        if(check_userID(random)){
            random = uuid.toString();
            continue;
        }
        else {
            break;
        }
    }
    return random;
}
```

```java
private boolean check_userID(String id){
    try {
        PreparedStatement prepStat = connect.getPrepstat( query: "SELECT user_id FROM login " +
                "WHERE user_id = ?");
        prepStat.setString( parameterIndex: 1, id);

        ResultSet myRs = prepStat.executeQuery();
        if(myRs.next()){
            return true;
        }
        else{
            return false;
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }
    return false;
}
```

This method will be the one to generate the random unique user ID. However, before it is registered as it is, it will first check whether the generated ID is already used or not. (it might not collide with other user, but just in case).

- Access Class

This class is to access the books table in the database which have the usual function such as viewing, insert and delete.

```java
public ResultSet showdata() {
    try {
        //to show the data based on the user
        PreparedStatement prepStat = connect.getPrepstat( query: "SELECT book_id, Title, Author, Extension FROM books " +
                "INNER JOIN storage ON books.book_id = storage.book_id_store " +
                "WHERE user_id = ?");
        prepStat.setString( parameterIndex: 1, login.get_userID());
        ResultSet myRs = prepStat.executeQuery();
        System.out.println("Data Showed!");
        return myRs;
    } catch (Exception e) {
        e.printStackTrace();
    }
    return null;
}
```

The code for showdata and searchdata is almost similar, the difference is the searchdata need the title parameter to search the data by title. The parameter user_id is needed so it only shows data which belong to the logged user.

```java
public void add_data(String title, String author, String file_loc){
    try {
        /*to initialize the insert query into the database
         * the 4 ? represent the 4 columns in the books database*/
        PreparedStatement prepStat = connect.getPrepstat( query: "insert into books " +
                "value(?,?,?,?)");

        //to store the blob file
        File file = new File(file_loc);
        InputStream inputStream = new FileInputStream(file);

        int random = random_number();

        prepStat.setInt( parameterIndex: 1, random);
        prepStat.setString( parameterIndex: 2, title);
        prepStat.setString( parameterIndex: 3, author);
        prepStat.setBlob( parameterIndex: 4, inputStream);

        int i = prepStat.executeUpdate();
        if (i > 0) {
            book.add_Book(random);
            JOptionPane.showMessageDialog( parentComponent: null, message: "Data is added");
        } else {
            JOptionPane.showMessageDialog( parentComponent: null, message: "Data is not saved");
        }
    } catch (FileNotFoundException e) {
        e.printStackTrace();
    } catch (SQLException e) {
        e.printStackTrace();
        JOptionPane.showMessageDialog( parentComponent: null, message: "The data is too big, please try again!");
    }
}
```

This method is for adding the data into the table. The parameter is from the information inputted by the user and the file and input stream is used to retrieve the blob file from the table to the selected location from the file chooser. The random is to randomize the new added book ID. After the data is successfully added, it will add the data in the storage table as well.

```java
public int random_number(){
    Random random = new Random();
    int id = random.nextInt(Integer.MAX_VALUE);
    /*to check if the id is used or not
     * if it is used, it will loop until it get the id that is not used*/
    while(true){
        if(checkid(id)){
            id = random.nextInt(Integer.MAX_VALUE);
            continue;
        }
        else break;
    }
    return id;
}
```

This will generate the random ID for the newly added book. Like the check_userID from the Register class, the checkid will check whether the ID had been taken or not, if not it will proceed, else, it will rerandomized again.

```java
public String browse_location(){
    JFileChooser chooser = new JFileChooser();
    chooser.showOpenDialog( parent: null);
    File f = chooser.getSelectedFile();
    String file = f.getAbsolutePath();

    return file;
}
```

This function will be used in the main user interface later for the file chooser, so the user can easily choose the file location instead of having to write the file path by themselves.

```java
public void retreive_data(int id, String file_loc) {
    if(book.check_book(id)) {
        try {
        /*to set the query to store the Loc if the title is equal to ?
        executing the query and storing the data of the Loc*/
            PreparedStatement prepStat = connect.getPrepstat( query: "SELECT Loc FROM books WHERE book_id=?");
            //to set the RTitle to the parameter of the query
            prepStat.setInt( parameterIndex: 1, id);

            ResultSet result = prepStat.executeQuery();

            // write binary stream into file
            File file = new File(file_loc);
            FileOutputStream output = new FileOutputStream(file);

            while (result.next()) {
                InputStream input = result.getBinaryStream( columnLabel: "Loc");          //to read the binary of the file in stream
                byte[] buffer = new byte[1024];
                while (input.read(buffer) > 0) {
                    output.write(buffer);
                }

                //to set the query to delete a row of value
                PreparedStatement pdsmt = connect.getPrepstat( query: "delete from books " +
                        "WHERE book_id = ?");
                pdsmt.setInt( parameterIndex: 1, id);
                pdsmt.executeUpdate();

                book.remove_book(id);
                JOptionPane.showMessageDialog( parentComponent: null,  message: "Book retrieved successfully. Writing to file " + file.getAbsolutePath());
            }
        } catch (SQLException e) {
            e.printStackTrace();
        } catch (FileNotFoundException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
```

In the retrieve method, it will retrieve the book by their ID, and it will ask the user where they want to write the file to, and output streamed the file to the file location. The inputStream is to read the binary(blob) file in stream, and the byte is to set the minimum file size, and then write it to the designated file path.

- Book Class

This class is to write the data in the storage table.

```java
public void add_Book(int book_id){
    try {
        PreparedStatement prepStat = connect.getPrepstat( query: "INSERT INTO storage " +
                "VALUES(?,?)");
        prepStat.setString( parameterIndex: 1, login.get_userID());
        prepStat.setInt( parameterIndex: 2, book_id);
        int i = prepStat.executeUpdate();
        if(i > 0){
            System.out.println("Data is added to storage");
        }
        else{
            System.out.println("Data failed to be added to storage");
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }
}
```

This method is to add the book data based on their ID to the table along with the logged user who inputted the data.

```java
public void remove_book(int book_id){
    try {
        PreparedStatement prepStat = connect.getPrepstat( query: "DELETE FROM storage " +
                "WHERE book_id_store = ?");
        prepStat.setInt( parameterIndex: 1, book_id);
        int i = prepStat.executeUpdate();
        //if the book from the storage table is deleted, it will delete the data in the book table as well
        if(i > 0){
            PreparedStatement pdstmt = connect.getPrepstat( query: "DELETE FROM books WHERE book_id = ?");
            pdstmt.setInt( parameterIndex: 1, book_id);
            pdstmt.executeUpdate();
            System.out.println("Data have been removed from storage");
        }
        else{
            System.out.println("Data failed to be removed from storage");
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }
}
```

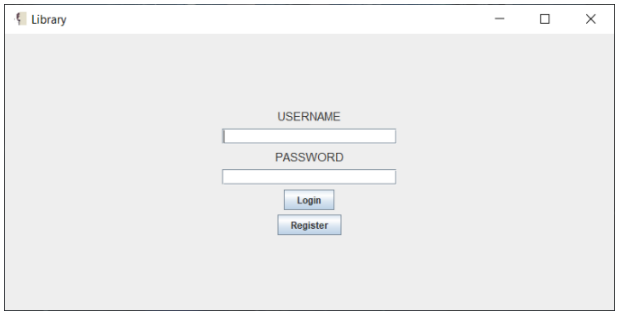
This method is used along side retrieve_data in the Access class, so when it retrieves the book file, delete the data from the books table and their data in the storage table as well.

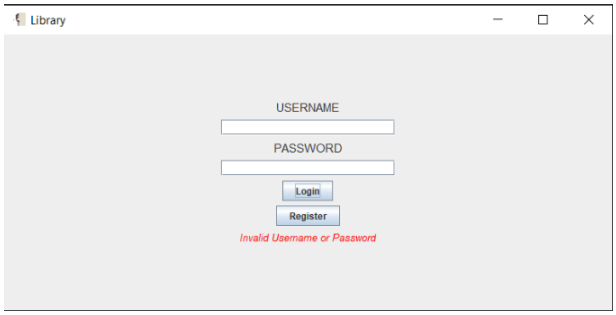
```java
public boolean check_book(int book_id){
    try {
        PreparedStatement prepStat = connect.getPrepstat( query: "SELECT * FROM storage " +
                "WHERE user_id = ? AND book_id_store = ?");
        prepStat.setString( parameterIndex: 1, login.get_userID());
        prepStat.setInt( parameterIndex: 2, book_id);

        ResultSet myRs = prepStat.executeQuery();
        if(myRs.next()){            //to check whether the book id is in the table or not
            return true;
        }
        else{
            return false;
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }
    return false;
}
```

This will be used in the retrieve_data in the Access class to first check whether the data user own the book or not, if yes then proceed, else, it will ask the user to confirm their inputted book ID to be retrieved.

IV. Evidence

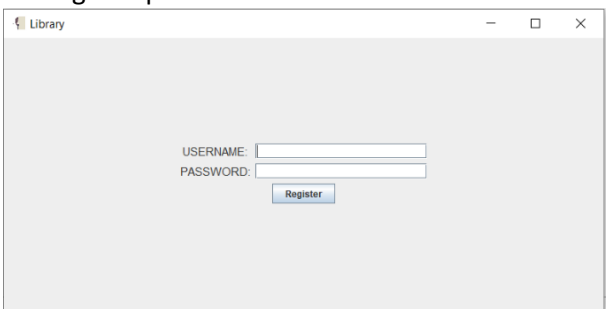
- Login panel

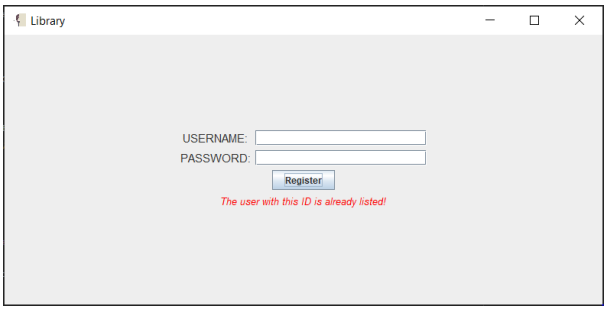




What appears when user inputted wrong username or password.

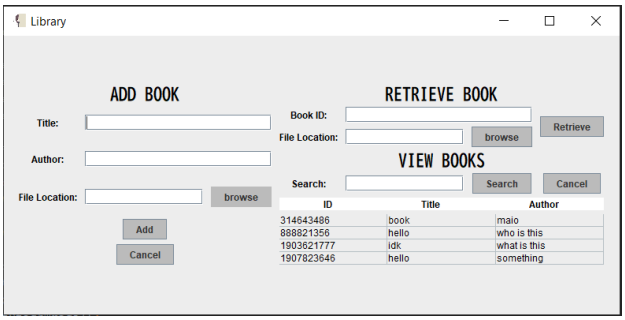
- Register panel





What appears when the registered account already registered

- Main panel



V. Reference

- https://youtu.be/2i4t-SL1VsU (how to make the database and connect the program with JDBC)
- https://www.youtube.com/watch?v=q8Z3CmruGzI (how to connect the GUI with the code for accessing the database)

- https://www.youtube.com/watch?v=7GZppdccFfs (how to insert data into Jtable)
- https://www.youtube.com/watch?v=ncOxOPRBUgM (how to search and update data in the Jtable)
- https://www.youtube.com/watch?v=IMjfmWVxd2E&t=472s (how to code the GUI manually)
- http://www.mysqltutorial.org/mysql-jdbc-blob (writing and retrieving blob file to and from database)