

# Secure Programming for Application Development Project

Professor- Peeyush Shankhareman

Name: Somesh Saxena

Student Number: x18176895

MSc. Cyber Security

National College of Ireland

2019-2020

## Content:

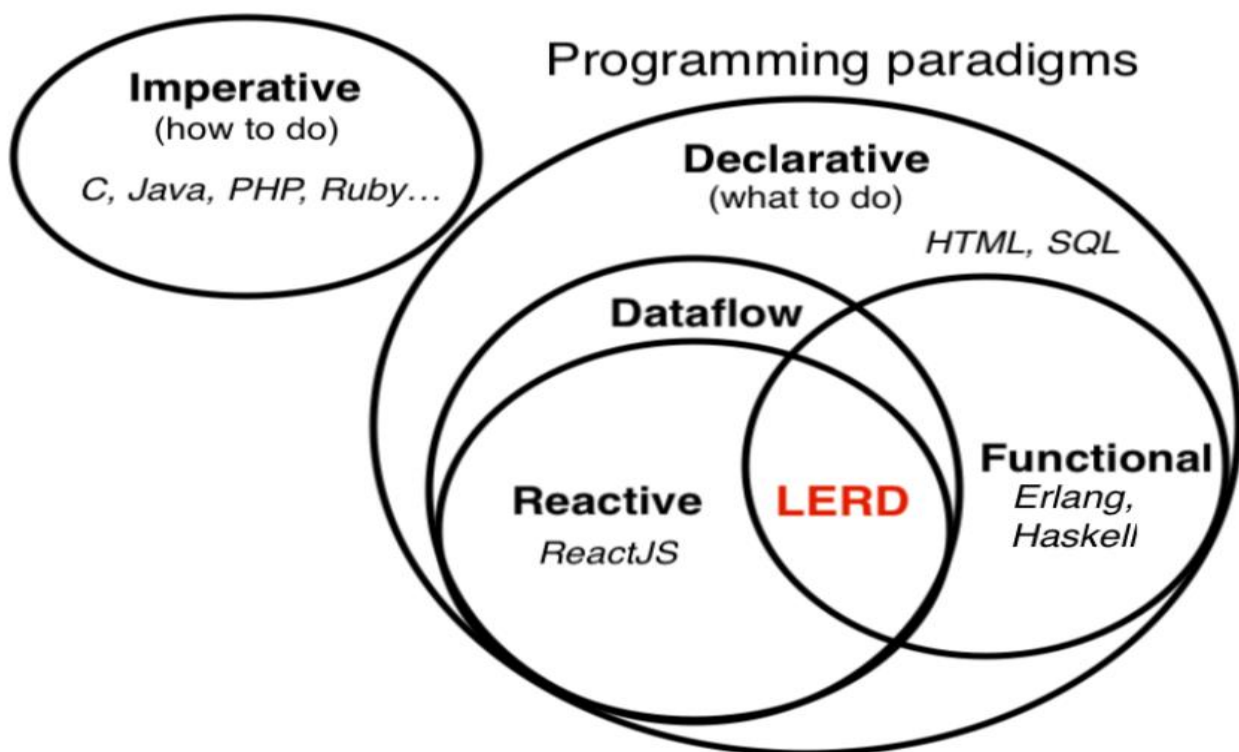
1. Introduction .....	3
2. Programming Paradigm .....	3
2.1 Imperative Programming Paradigm .....	4
2.2 Declarative Programming Paradigm .....	5
3. Security Testing .....	6
4. Application Testing .....	7
5. Proposed Solution .....	19
6. Conclusion .....	19
References .....	20

## I. Introduction:

The difficulty level of an program has been raised due to the dramatic growth in the field of technology. This has been a crucial and essential obstacle in monitoring of applications. Any compliance monitoring security testing is directed primarily at retaining CIA traits. Secure coding has become an important practice in software development to remedy defects than if they were discovered earlier in the development lifecycle or all together prevented. The testing tests whether any of the malicious data is vulnerable to a given program. As security vulnerabilities grow, new testing models are required that can help minimize all of these vulnerabilities.

## II. Programming Paradigm:

Every programming language is subject to at least one programming standards. Each programming paradigm contains a lot of ideas about programming. A methodology requires programming styles, templates, structures, and instruction's to tell a programmer which way to do programming. Such paradigms primarily include consequences for the language's execution model such as enabling side effects, or how the execution model determines the sequence of operations. Many paradigms deal primarily with the way code is ordered.[1] Implementing a model includes the variations in the way computing constructs are interpreted and represented by the systems and technology used [2].



Low Entropy Reactive Design

In this section, there are numerous programming approaches we have been examining.

## 1. Imperative Programming Paradigm:

In the paradigm the CPU instructions are imperative. This paradigm is necessary if your compilers, interpreters that communicate with the low level code, are created. It changes state by assignments, through executing tasks step by step. The languages only consist of simple controls like loops, conditions, etc. Examples are COBOL, Assembly language, FORTRAN, C, C++, Java, etc.

The imperative paradigms were used in the early stages of programming languages such as software's, assembly and high level languages and how the role of the imperative programming paradigm plays a crucial role in the modern field of computer science. A software that is written using imperative programming paradigm has a sequence of commands and statements that run on memory-save data. Imperative theory uses the method of side effects performed by declarations of selections. The Imperative programming paradigm concept was based on the von Neumann stored program model. It is explained how imperative programming model can help computer science students learn programming language fundamentals which can further help them learn and understanding of other high-level languages better.

## Procedural Programming Paradigm:

Procedural programming uses small, reusable blocks of codes which consist of control flow statements that change the state of the program. The modular blocks are referred to as procedures, subroutines of sequence control flow statements. Procedure-oriented programming is the foundation of all programming and once a developer knows process-oriented programming he will work on any programming. Within each procedure, local variables are used without affecting the outside procedure of the programme.[3]

Although the procedure-oriented programming does have some disadvantages.

- There is no modularity in the code and the entire system is revealed at once with no encryption or information shielding.
- Due to the emphasis on instructions and not data itself, it is often difficult to coordinate real life issues.
- Code does not contain a specific hierarchy, and each program is treated equally.

## A. Object-Oriented Programming Paradigm:

In object-oriented the methods or procedures are written in classes controlling the actions of these methods by sending messages during the development and use of the examples. It has four fundamental principles like abstraction, encapsulation, inheritance and polymorphism. These principles allow for effective object-oriented programming from the perspective of security. Data is encapsulated which are called as objects and can only be called through methods.

Object-oriented theory as a computational structure, referred to as an entity model focused on the convergence of abstraction, encapsulation, modularity and hierarchy by the idea of competition and continuity.[4]

He also offered the Object-oriented design benefit over procedural programming and they are:

- Object-oriented architecture helps in the creation of dynamic programs.

- The pattern allows all programming and architecture to be reused.
- The framework based on opposed criteria is scalable and can be built over time as needed.

In [5] the reviewer performed a systematic comparative examination of the merits of the approaches and languages of object-oriented and traditional procedural software architecture. The test case was tested on 40 topics, and the data was obtained using procedural and object-oriented model approaches. Using object-oriented programming was the product of the study and even easier than the approach of recursive programming.

### **B. Parallel Processing Programming Paradigm:**

Different parts of the program executes various functions in concurrent and cooperative fashion in the parallel programming. Parallelism can be seen in data structure, however, this kind of parallelism makes system execution in parallel and is similar except on different data pieces. [6] It ensures that the execution of program instructions is split between several processes, thereby providing the advantage of running the program in less time than the convectional imperative programming model. Examples of parallel processing strategies are NESL and C/C++ .

## **2. Declarative Programming Paradigm:**

The language of declarative programming is a language written application. The declarative programming is elevated from the coder viewpoint to a higher degree of abstraction. The programmer must specify what needs to be measured at this stage rather than how it should be measured. As for the Kowalski algorithm = logic + control. Declarative programming has two meanings: Strong sense and Weak sense. Strong meaning, the programmer has only to include logic in algorithm and all other control knowledge is automatically supplied by the system. In weak sense meaning that the programmer must send control knowledge apart from logic to order to achieve effective performance.[7] Under Declarative programming paradigm Logic, Functional, Database programming comes.

### **A. Logic Programming Paradigm:**

The Logic programming model is an abstract computing model which is used to solve problem like puzzles. In the logic programming the programmer has some information of the foundation which is then supplied to the computer and on basis the output is generated.

The following characteristics the logic programming is distinct from other programming:

- Computing outstands in all respects across the world.
- Values are allocated to variables automatically through the general unifier.
- Control is provided by single mechanism.

### **B. Functional Programming Paradigm:**

Functional programming is an example of declarative programming which uses mathematical function computation without changing the state and mutating data. The order in which functions are executed doesn't matter because they don't have state and all functions work independently. Changing execution order would have the same performance. The languages like Perl, JS uses this paradigm.

Security testing such as debugging is easier because pure functions, state does not matter. The programs are developed from function descriptions in the functional programming paradigm. Usually the program is mapped to a variant of  $\lambda$ - calculus and is then distributed as denotations. There are two strict functional languages such as Standard ML and a functional language such as Haskell.

### III. Security Testing: -

Different attacks were carried out on the network in order to access sensitive information. This is attributed to a lack of technical knowledge and a lack of awareness of security issues in software applications. Specification of security requirements is necessary in order to improve protection, but it is a challenging task. SDL does this, where a message goes through different stages and encryption is checked at each point.

Security testing can be done in various forms across the SSDLC.

SDLC Phase	Security Testing Process
<b>Requirements</b>	Reviewing policies and procedures and creation of abuse cases
<b>Design</b>	Risk analysis for each requirement and creation of test plan including security tests
<b>Coding and Unit testing</b>	Static Analysis (Whitebox Testing) and Dynamic Security testing (Blackbox Testing)
<b>Integration Testing</b>	Black Box testing and Threat Modelling
<b>System Testing</b>	Vulnerability Scanning
<b>Implementation</b>	Penetration Testing and vulnerability scanning
<b>Support</b>	Impact Analysis of Regular patches and updates

Below are the various methodologies of security testing which can aid in application testing.

In penetration testing methodologies: A comparison and evaluation [8] author compares various pen testing frameworks such as OSSTMM, ISSAF, OWASP, MSF, BSIMM and PTES in order to evaluate if the system for pen-testing is really a framework.

- i. **ISSAF:** ISSAF stands for Information System Security Assessment Framework by OISSG. It encapsulates various methodologies in this setting, which aims to cover all potential areas of penetration testing. The penetration monitoring that comes under this structure is divided primarily into three phases: planning & preparation, assessment, and reporting & clean-up. One of ISSAF benefits is that there is a clear relationship between the assignments and their corresponding tools.
- ii. **OSSTMM:** - OSTMM stands for open source security testing methodology introduced by ISECOM. It was developed under peer review and was open source, but this application's new release is paid. The encapsulated modules and channels are in OSSTMM methodologies. OSSTMM is essentially an auditing technique that is used to meet the regulatory criteria for corporate properties provided it is identified to the security auditor.
- iii. **OWASP:** OWASP provides various open source tools. The OWASP testing Guide (OTG) is comprised of three sections: the web application development, the web application testing and reporting. The methodologies in application can be used separately or can be paired with software system.
- iv. **BSIMM:** The major operations of BSIIMM are 112 which are split into 12 parts which serve 4 areas, primarily intelligence, SSDL touch points, and deployment. It does not decide what methods to use and how relative to other methodologies, but instead describes the strategies that are appropriate for organizations.
- v. **PTES:** PTES stands for Penetration Testing Execution Standards is a penetration testing standard created in 2009. It consists of pre- engagement level, collection of information,

modelling of risks, analysis of vulnerability, exploitation, post exploitation and reporting. Inside it PTES uses other structures rather than relying on others.

- vi. MSF: Metasploit Framework is a suite of software designed to recognise exploit vulnerabilities for pen-testing and intrusion detection. MSF is ideal for all technology experts who have a strong knowledge of pen-testing and have been command on command lines. Compared to ISSAF and OSSTMM, Metasploit is a functional approach that offers resources that need to be implemented, rather than procedure papers.

In this paper [9] reviewer proposed a new penetration testing tool called PJCT to test java code. Which checks any given java code major security attribute. The PJCT can detect seven attributes present or missing. The tool was designed using the following 7 attributes, and the following are:

- Inclusion of Network and other security packages.
- Exception handling techniques.
- Establishment of the open and close session.
- Transmission of the Data packets in secure mode.
- Cookies creation.
- Garbage Collection.
- Multithreading

In paper [10] by authors conducted a comparative analysis of various static code analysis tools based on standards for java, such as 10 rule and JPL coding standards. The research was carried out on tools like Squale, Sonar, Eval metrics, Jlint and Findbugs, and the results of the study were provided for each tool. In addition to the analysis, the tools were also classified in 7 separate categories based on the enforces tool. These seven rules are Style, Concurrency, Exception, Quality, Security, Dependency, and Compatibility. In conclusion, the authors claimed that they have their own features and should be used according to the code analysis.

#### IV. Application Testing: -

There are things that are preserved by security testing and they are:

- Authentication
- Authorization
- Availability
- Confidentiality
- Integrity
- Non-repudiation.

Security prevents software's against potential malware and other unexpected threats which may lead to malfunctioning or misuse. Those attacks can be malicious or unplanned. Security monitoring tools evaluate whether third party requests are helpful or harmful.

##### A. Well Know Testing Methods: -

###### i. Static Code Analysis: -

Analysis of static code includes evaluating code and finding bugs that may result in violation of security in programs or in resources accessed by application. Example the type of coding error that can cause buffer overflow or attack by SQL injection. In Static code analysis, however, the source code is analyzed using an automated tool, static code analysis can be done by hand, but it is not possible for developers to manually go through the code and find large-scale security flaws and a developer may miss major security flaws.



ii. Dynamic Code Analysis: -

Analysis of complex programming is a technique for evaluating the program at execution. Dynamic analysis will break into several stages:

- Preparation of data
- Running test program
- Gathering information
- Analysing the output data

There are also dynamic analytics tool, and each tool differs in program interaction. Dynamic analysis is done by forwarding a series of data to a system to be searched for. Therefore, analytical performance depends on the consistency and quantity of data from the input samples.

iii. White Box Testing: -

In this testing methodology the research is performed on the software's internal framework, architecture and coding. This focuses primarily on checking the application's input and output and also on improving architecture & usability, reinforcing security. It is one of the component Box application software testing methods and its equivalent is Blackbox testing involving public or end user testing.

Following are testing involved for white box testing:

- Internal security holes.
- Poorly structured coding processes.
- Flow of specific inputs through the code.
- Expected output.
- Functionality of conditional loops.
- Testing of each statement, object and functions.

S.N.	Black Box Testing	Grey Box Testing	White Box Testing
1	The Internal Workings of an application are not required to be known	Somewhat knowledge of the internal workings are known	Tester has full knowledge of the Internal workings of the application
2	Also known as closed box testing, data driven testing and functional testing	Another term for grey box testing is translucent testing as the tester has limited knowledge of the insides of the application	Also known as clear box testing, structural testing or code based testing
3	Performed by end users and also by testers and developers	Performed by end users and also by testers and developers	Normally done by testers and developers
4	Testing is based on external expectations - Internal behavior of the application is unknown	Testing is done on the basis of high level database diagrams and data flow diagrams	Internal workings are fully known and the tester can design test data accordingly
5	This is the least time consuming and exhaustive	Partly time consuming and exhaustive	The most exhaustive and time consuming type of testing
6	Not suited to algorithm testing	Not suited to algorithm testing	Suited for algorithm testing
7	This can only be done by trial and error method	Data domains and Internal boundaries can be tested, if known	Data domains and Internal boundaries can be better tested

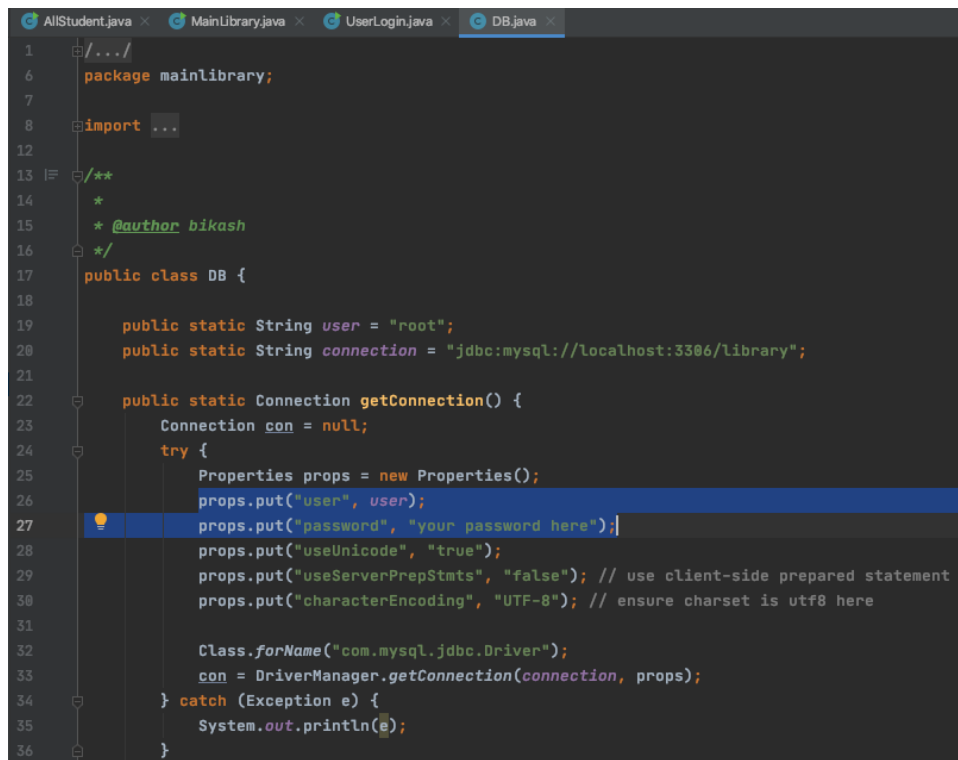


## B. Manual Code Review: -

File Name	Line(s) number	Security Issue	Fix Description
DB.java	19,20	The public variable is not kept final, and this portion of code allows for malicious code to be accessed.	All public variable should be retained as final. If proper sanitization procedures are to be enforced.
DB. Java	34	The Exceptions were found even though they were not thrown out.	All the Exception tossed should be caught or even capture runtime.
UserViewBook.java	255	Wrong operator used to evaluate the strings.	Use of equal() method or compareTo() should be used to compare the strings.
LibrarianDao.java	47	We pass Non-constant string to execute a process.	Use prepared declaration to shield code from SQL injection.
LibrarianSucess.java	383	The system fails to close database connection.	Database connection should always be closed.
UserDao.java	23	The attacker could exploit SQL query.	Proper Sanitization of input data and prepared statement should be used.
UserDao.java	57	The code is vulnerable for XSS attack.	Comprehension of all potential inputs where untrusted inputs may enter software.
LibrarianDao.java	41	Storing passwords in cleartext format in database.	Password needs to be hashed.

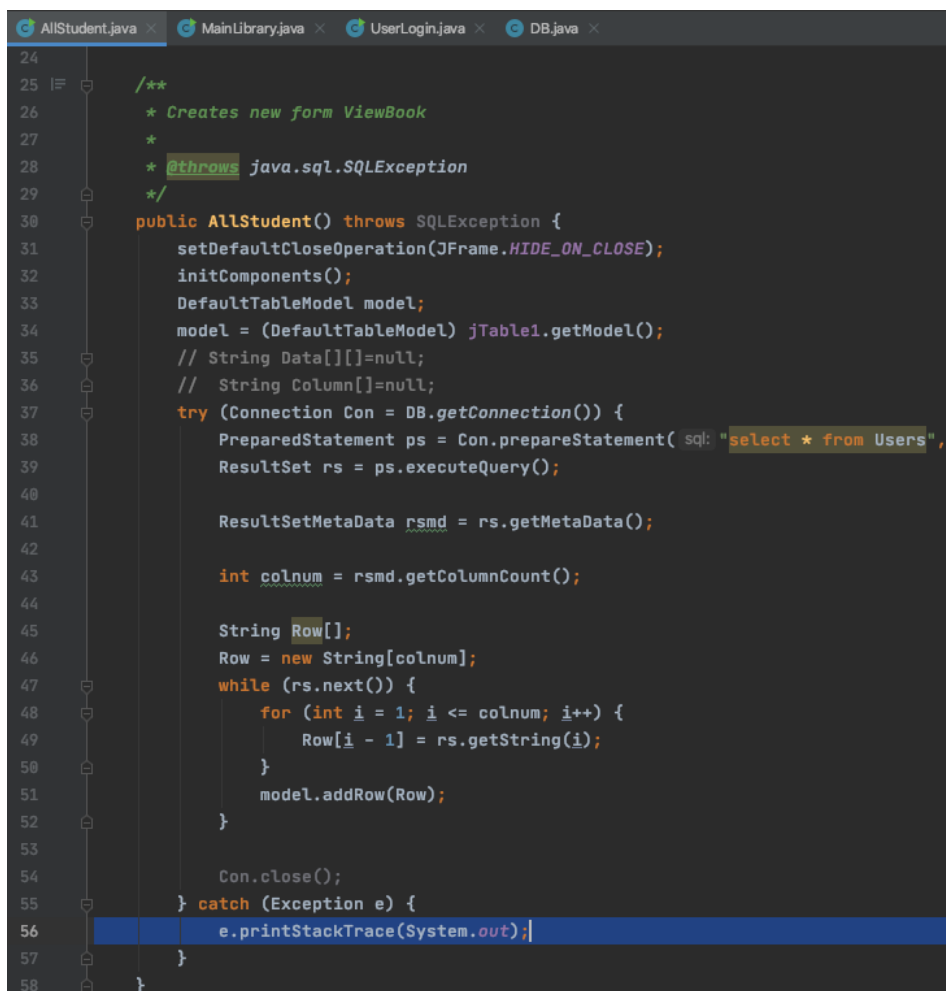
## Screenshots of reported vulnerabilities:

### 1. Cleartext user credentials in the code:



```
1  .../
6  package mainlibrary;
7
8  import ...
12
13  /**
14   *
15   * @author bikash
16   */
17  public class DB {
18
19      public static String user = "root";
20      public static String connection = "jdbc:mysql://localhost:3306/library";
21
22      public static Connection getConnection() {
23          Connection con = null;
24          try {
25              Properties props = new Properties();
26              props.put("user", user);
27              props.put("password", "your password here");
28              props.put("useUnicode", "true");
29              props.put("useServerPrepStmts", "false"); // use client-side prepared statement
30              props.put("characterEncoding", "UTF-8"); // ensure charset is utf8 here
31
32              Class.forName("com.mysql.jdbc.Driver");
33              con = DriverManager.getConnection(connection, props);
34          } catch (Exception e) {
35              System.out.println(e);
36          }
37      }
38  }
```

### 2. Print stack trace error handling:



```
24
25  /**
26   * Creates new form ViewBook
27   *
28   * @throws java.sql.SQLException
29   */
30  public AllStudent() throws SQLException {
31      setDefaultCloseOperation(JFrame.HIDE_ON_CLOSE);
32      initComponents();
33      DefaultTableModel model;
34      model = (DefaultTableModel) jTable1.getModel();
35      // String Data[][]=null;
36      // String Column[]=null;
37      try (Connection Con = DB.getConnection()) {
38          PreparedStatement ps = Con.prepareStatement("select * from Users");
39          ResultSet rs = ps.executeQuery();
40
41          ResultSetMetaData rsmd = rs.getMetaData();
42
43          int colnum = rsmd.getColumnCount();
44
45          String Row[];
46          Row = new String[colnum];
47          while (rs.next()) {
48              for (int i = 1; i <= colnum; i++) {
49                  Row[i - 1] = rs.getString(i);
50              }
51              model.addRow(Row);
52          }
53
54          Con.close();
55      } catch (Exception e) {
56          e.printStackTrace(System.out);
57      }
58  }
```

### 3. Absence of input validation on Username and Password:

```
AllStudent.java x LibrarianDao.java x MainLibrary.java x UserLogin.java x DB.java x
25 }
26
27 public static int delete(int id) {
28     int status = 0;
29     try {
30         Connection con = DB.getConnection();
31         PreparedStatement ps = con.prepareStatement( sql: "delete from Librarian where id=?");
32         ps.setInt( parameterIndex: 1, id);
33         status = ps.executeUpdate();
34         con.close();
35     } catch (Exception e) {
36         System.out.println(e);
37     }
38     return status;
39 }
40
41 public static boolean validate(String name, String password) {
42     boolean status = false;
43     try {
44         Connection con = DB.getConnection();
45         String select = "select * from Librarian where UserName= '" + name + "' and Password='" + password + "'";
46         Statement selectStatement = con.createStatement();
47         ResultSet rs = selectStatement.executeQuery(select);
48
49         status = rs.next();
50         con.close();
51     } catch (Exception e) {
52         System.out.println(e);
53     }
54 }
```

### 4. Application storing password in cleartext format:

```
AllStudent.java x LibrarianDao.java x MainLibrary.java x UserLogin.java x DB.java x
23 }
24     return status;
25 }
26
27 public static int delete(int id) {
28     int status = 0;
29     try {
30         Connection con = DB.getConnection();
31         PreparedStatement ps = con.prepareStatement( sql: "delete from Librarian where id=?");
32         ps.setInt( parameterIndex: 1, id);
33         status = ps.executeUpdate();
34         con.close();
35     } catch (Exception e) {
36         System.out.println(e);
37     }
38     return status;
39 }
40
41 public static boolean validate(String name, String password) {
42     boolean status = false;
43     try {
44         Connection con = DB.getConnection();
45         String select = "select * from Librarian where UserName= '" + name + "' and Password='" + password + "'";
46         Statement selectStatement = con.createStatement();
47         ResultSet rs = selectStatement.executeQuery(select);
48     }
49 }
```

## 5. Password complexity not enforced:

```
ident.java x LibrarianDao.java x MainLibrary.java x UserLogin.java x DB.java x
package mainlibrary;

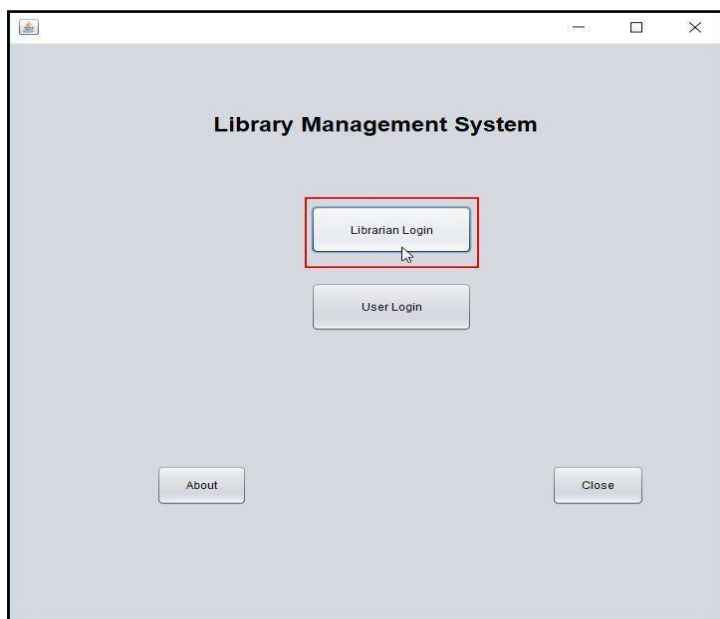
import java.sql.*;

public class LibrarianDao {

    public static int save(String name, String password, String email, String address, String city, String contact) {
        int status = 0;
        try {

            Connection con = DB.getConnection();
            PreparedStatement ps = con.prepareStatement( sql: "insert into librarian(name,password,email,address,city,contact) values(?,?,?,?,?,?)" );
            ps.setString( parameterIndex: 1, name);
            ps.setString( parameterIndex: 2, password);
            ps.setString( parameterIndex: 3, email);
            ps.setString( parameterIndex: 4, address);
            ps.setString( parameterIndex: 5, city);
            ps.setString( parameterIndex: 6, contact);
            status = ps.executeUpdate();
            con.close();
        } catch (Exception e) {
            System.out.println(e);
        }
    }
}
```

## 6. SQL injection on login page:



Payload used: 'or' 1=1--

The screenshot shows a web application window titled "Librarian Login". It contains two input fields: "Username" with the value "Zero" and "Password" with masked characters "\*\*\*\*\*". A red rectangular box highlights these two fields. Below the password field, a blue callout bubble contains the text "'or' 1=1--". To the right of the input fields is a "Login" button, and below it is a "Back" button.

Successful login into the application.

The screenshot shows a web application window titled "Librarian Dashboard". It contains several input fields and buttons. On the left, there are four input fields labeled "Name", "Library ID", "Email", and "Contact No.". To the right of these fields are five buttons: "View Issued Books", "Issue Book", "Return Book", "Delete Book", and "Add Student". Below the input fields are two buttons: "Add Book" and "View Book". At the bottom center is a "LogOut" button. A red rectangular box highlights the entire dashboard area.

## C. Testing Methodology Used: -

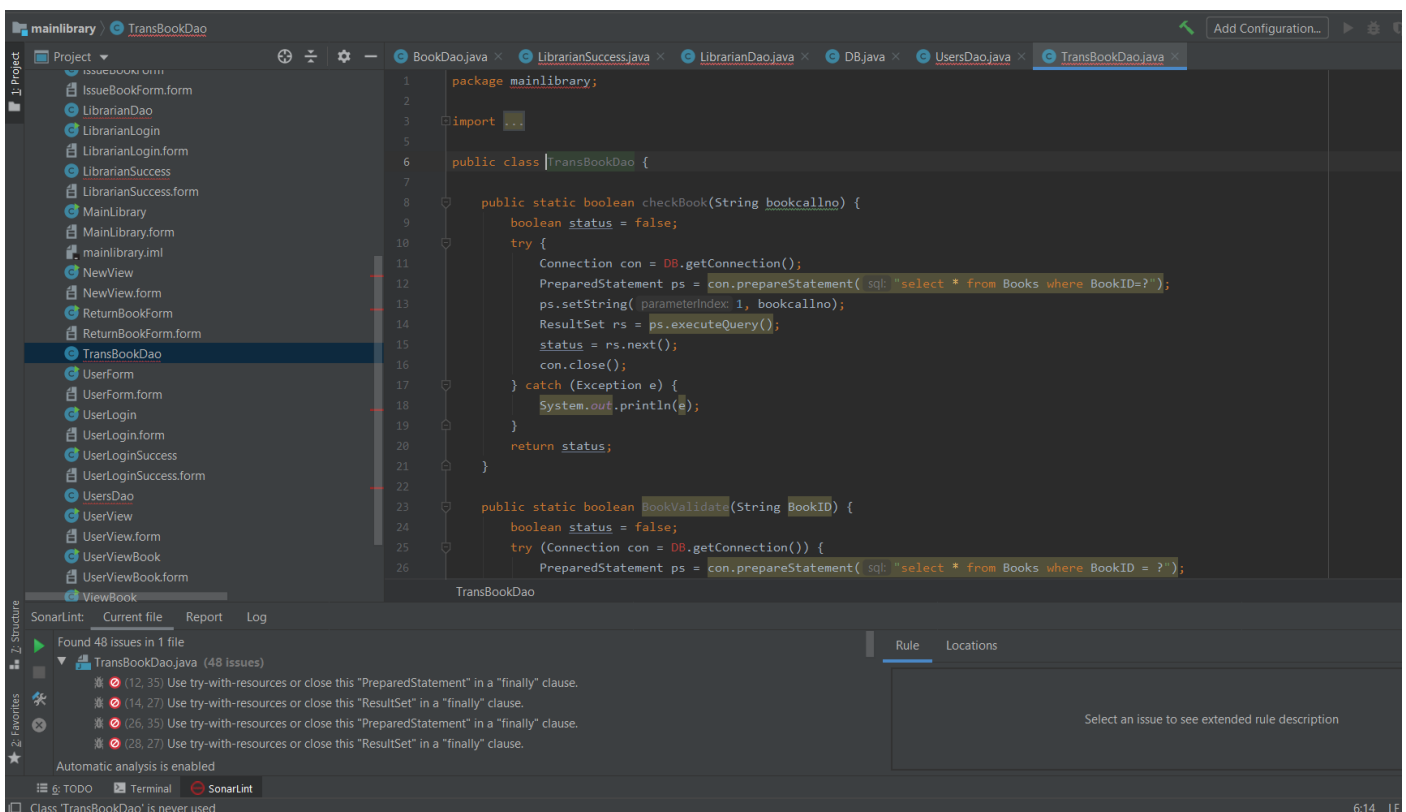
For static analysis we have used two tools.

- Sonar Lint.
- Visual Code Grepper.

### 1) Sonar Lint:

Sonarlint is a plugin for IDE's such as IntelliJ and eclipse that provides programmers with input on their code and consistency concerns regarding the error.

Below are the screenshots in IntelliJ IDE with Sonarlint plugin that we took for source code.



Project ▾

- IssueBookForm.form
- LibrarianDao
- LibrarianLogin
- LibrarianLogin.form
- LibrarianSuccess
- LibrarianSuccess.form
- MainLibrary
- MainLibrary.form
- mainlibrary.iml
- NavigationView
- NavigationView.form
- ReturnBookForm
- ReturnBookForm.form
- TransBookDao
- UserForm
- UserForm.form
- UserLogin
- UserLogin.form
- UserLoginSuccess
- UserLoginSuccess.form
- UsersDao
- UIView
- UIView.form
- UIViewBook
- UIViewBook.form

DB.java

```
1  //.../
6  package mainlibrary;
7
8  import ...
12
13  /**
14   * @author bikash
15   */
16
17  public class DB {
18
19      public static String user = "root";
20      public static String connection = "jdbc:mysql://localhost:3306/library";
21
22      public static Connection getConnection() {
23          Connection con = null;
24          try {
25              Properties props = new Properties();
26              props.put("user", user);
27              props.put("password", "");
28              props.put("useUnicode", "true");
29              props.put("useServerPrepStmts", "false"); // use client-side prepared statement
30              props.put("characterEncoding", "UTF-8"); // ensure charset is utf8 here
31
32              Class.forName("com.mysql.jdbc.Driver");
```

SonarLint: Current file Report Log

Found 8 issues in 1 file

DB.java (8 issues)

- ✖ (17, 13) Add a private constructor to hide the implicit public one.
- ✖ (32, 18) Remove this "Class.forName()", it is useless.
- ✖ (35, 12) Replace this use of System.out or System.err by a logger.
- ⚠ (10, 0) Remove this unused import 'java.sql.SQLException'.

Automatic analysis is enabled

Rule Locations

Select an issue to see extended rule description

g: TODO Terminal SonarLint

Project ▾

- IssueBookForm.form
- LibrarianDao
- LibrarianLogin
- LibrarianLogin.form
- LibrarianSuccess
- LibrarianSuccess.form
- MainLibrary
- MainLibrary.form
- mainlibrary.iml
- NavigationView
- NavigationView.form
- ReturnBookForm
- ReturnBookForm.form
- TransBookDao
- UserForm
- UserForm.form
- UserLogin
- UserLogin.form
- UserLoginSuccess
- UserLoginSuccess.form
- UsersDao
- UIView
- UIView.form
- UIViewBook
- UIViewBook.form

UsersDao.java

```
26      status = rs.next();
27      con.close();
28  } catch (Exception e) {
29      System.out.println(e);
30  }
31  return status;
32  }
33
34  public static boolean checkIfAlready(String UserName) {
35      boolean status = false;
36      try {
37          Connection con = DB.getConnection();
38          String select = "select * from Users where UserName= '" + UserName + "'";
39          Statement selectStatement = con.createStatement();
40          ResultSet rs = selectStatement.executeQuery(select);
41          status = rs.next();
42          con.close();
43      } catch (Exception e) {
44          System.out.println(e);
45      }
46      return status;
47  }
48
49  public static int AddUser(String User, String UserPass, String UserEmail, String Date) {
50      UsersDao > CheckIfAlready()
```

SonarLint: Current file Report Log

Found 16 issues in 1 file

UsersDao.java (16 issues)

- ✖ (24, 40) Use try-with-resources or close this "Statement" in a "finally" clause.
- ✖ (25, 27) Use try-with-resources or close this "ResultSet" in a "finally" clause.
- ✖ (39, 40) Use try-with-resources or close this "Statement" in a "finally" clause.
- ✖ (40, 27) Use try-with-resources or close this "ResultSet" in a "finally" clause.

Automatic analysis is enabled

Rule Locations

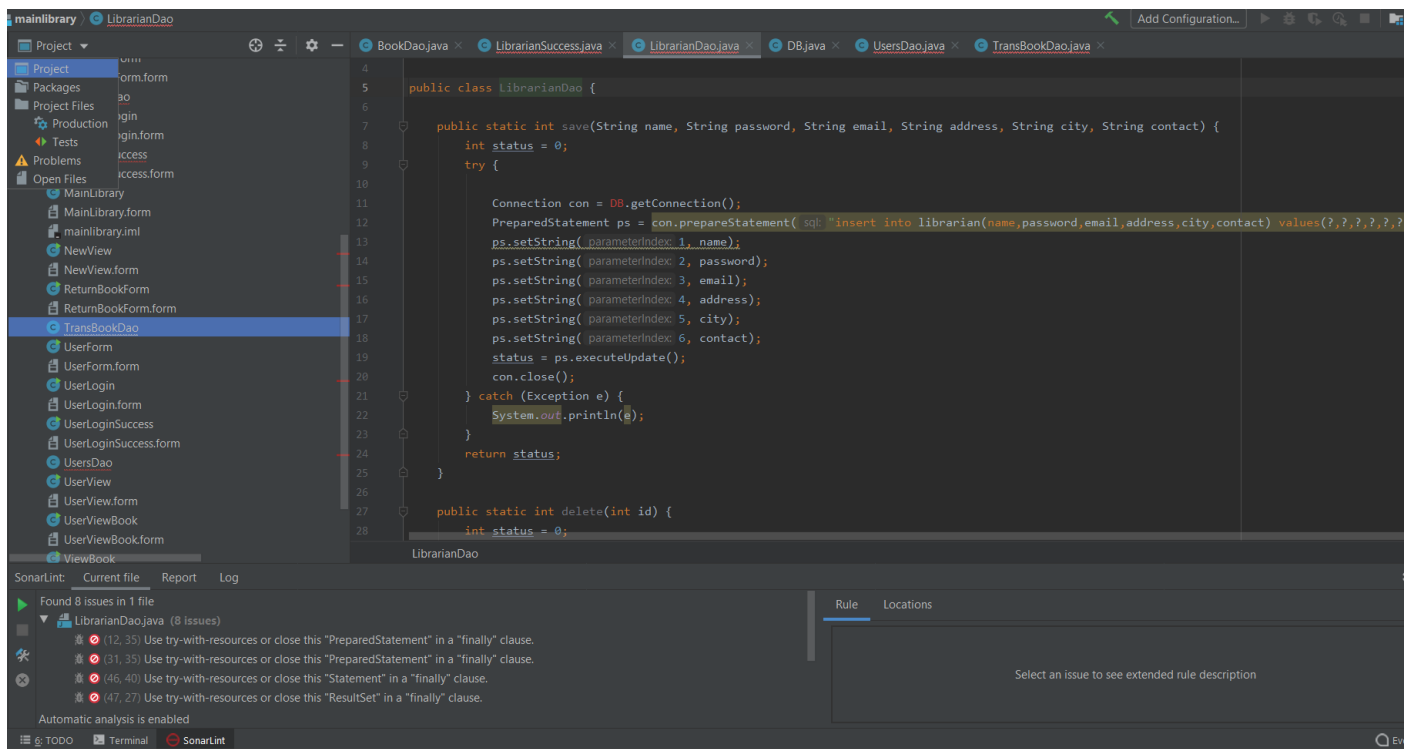
Select an issue to see extended rule description

g: TODO Terminal SonarLint

No data sources are configured to run this SQL and provide advanced code assistance. Disable this inspection via problem menu (Alt+Enter). SQL dialect is not configured.

38:32





As we can see from the above images, Sonarlint outlined the issue on the code and also explains what the problem is. Visual studio grepper (VSG) is another method we used for static analysis.

## 2) Visual Code Grepper: -

Visual Code Grepper is an automated security testing tool that can work with C, C++ java Visual Basics and PL/SQL. It has apps that help conduct security evaluation. Besides carrying out complicated security tests, it also has config file for each programming language that helps the user to add some function or scan for some function. It also offers text or graphical format production and distinguishes buffer overflow and signed/unsigned integers as well.

Target Files Results Summary Table

**POTENTIAL ISSUE: Potentially Unsafe Code - Public Class Not Declared as Final**  
 Line: 7 - C:\Users\Mandar\Desktop\SPAD\_Project\trainingProject\securityProject\src\mainlibrary  
 \BookDao.java  
 The class is not declared as final as per OWASP recommendations. It is considered best practice to make classes final where possible and practical (i.e. It has no classes which inherit from it). Non-Final classes can allow an attacker to extend a class in a malicious manner. Manually inspect the code to determine whether or not it is practical to make this class final.  
 public class BookDao {

**POTENTIAL ISSUE: Potentially Unsafe Code - Public Class Not Declared as Final**  
 Line: 17 - C:\Users\Mandar\Desktop\SPAD\_Project\trainingProject\securityProject\src\mainlibrary  
 \BookForm.java  
 The class is not declared as final as per OWASP recommendations. It is considered best practice to make classes final where possible and practical (i.e. It has no classes which inherit from it). Non-Final classes can allow an attacker to extend a class in a malicious manner. Manually inspect the code to determine whether or not it is practical to make this class final.  
 public class BookForm extends javax.swing.JFrame {

**SUSPICIOUS COMMENT: Comment Indicates Potentially Unfinished Code -**  
 Line: 212 - C:\Users\Mandar\Desktop\SPAD\_Project\trainingProject\securityProject\src\mainlibrary  
 \BookForm.java  
 TODO add your handling code here:

**SUSPICIOUS COMMENT: Comment Indicates Potentially Unfinished Code -**  
 Line: 216 - C:\Users\Mandar\Desktop\SPAD\_Project\trainingProject\securityProject\src\mainlibrary  
 \BookForm.java  
 TODO add your handling code here:

**SUSPICIOUS COMMENT: Comment Indicates Potentially Unfinished Code -**  
 Line: 220 - C:\Users\Mandar\Desktop\SPAD\_Project\trainingProject\securityProject\src\mainlibrary  
 \BookForm.java  
 TODO add your handling code here:

**SUSPICIOUS COMMENT: Comment Indicates Potentially Unfinished Code -**  
 Line: 224 - C:\Users\Mandar\Desktop\SPAD\_Project\trainingProject\securityProject\src\mainlibrary

Language: Java File Suffixes: .java|.jsp|.jspx|web.xml|config.xml [21 Files]

Target Files Results Summary Table

**SUSPICIOUS COMMENT: Comment Indicates Potentially Unfinished Code -**  
 Line: 225 - C:\Users\Mandar\Desktop\SPAD\_Project\trainingProject\securityProject\src\mainlibrary  
 \AllStudent.java  
 TODO add your handling code hereset

**SUSPICIOUS COMMENT: Comment Indicates Potentially Unfinished Code -**  
 Line: 230 - C:\Users\Mandar\Desktop\SPAD\_Project\trainingProject\securityProject\src\mainlibrary  
 \AllStudent.java  
 TODO add your handling code here:

**SUSPICIOUS COMMENT: Comment Indicates Potentially Unfinished Code -**  
 Line: 234 - C:\Users\Mandar\Desktop\SPAD\_Project\trainingProject\securityProject\src\mainlibrary  
 \AllStudent.java  
 TODO add your handling code here:

**SUSPICIOUS COMMENT: Comment Indicates Potentially Unfinished Code -**  
 Line: 240 - C:\Users\Mandar\Desktop\SPAD\_Project\trainingProject\securityProject\src\mainlibrary  
 \AllStudent.java  
 TODO add your handling code here:

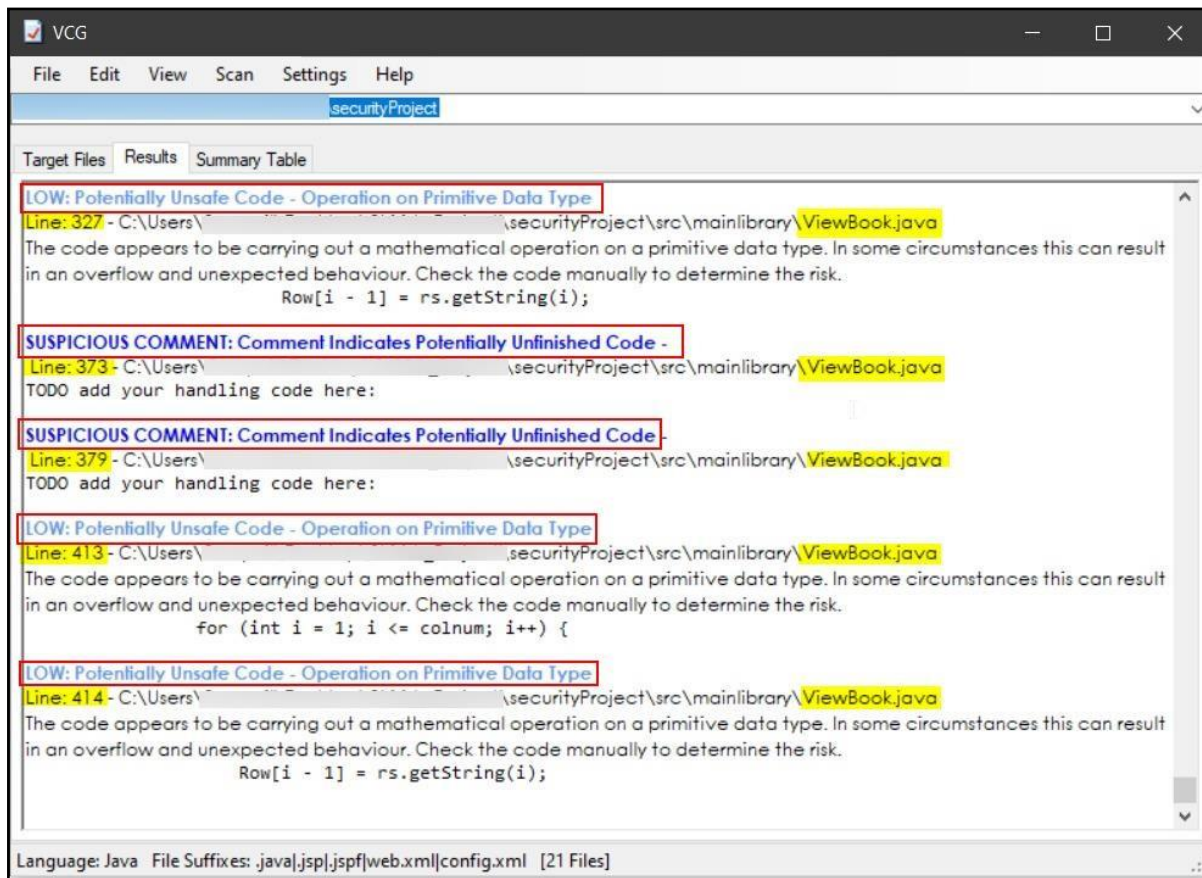
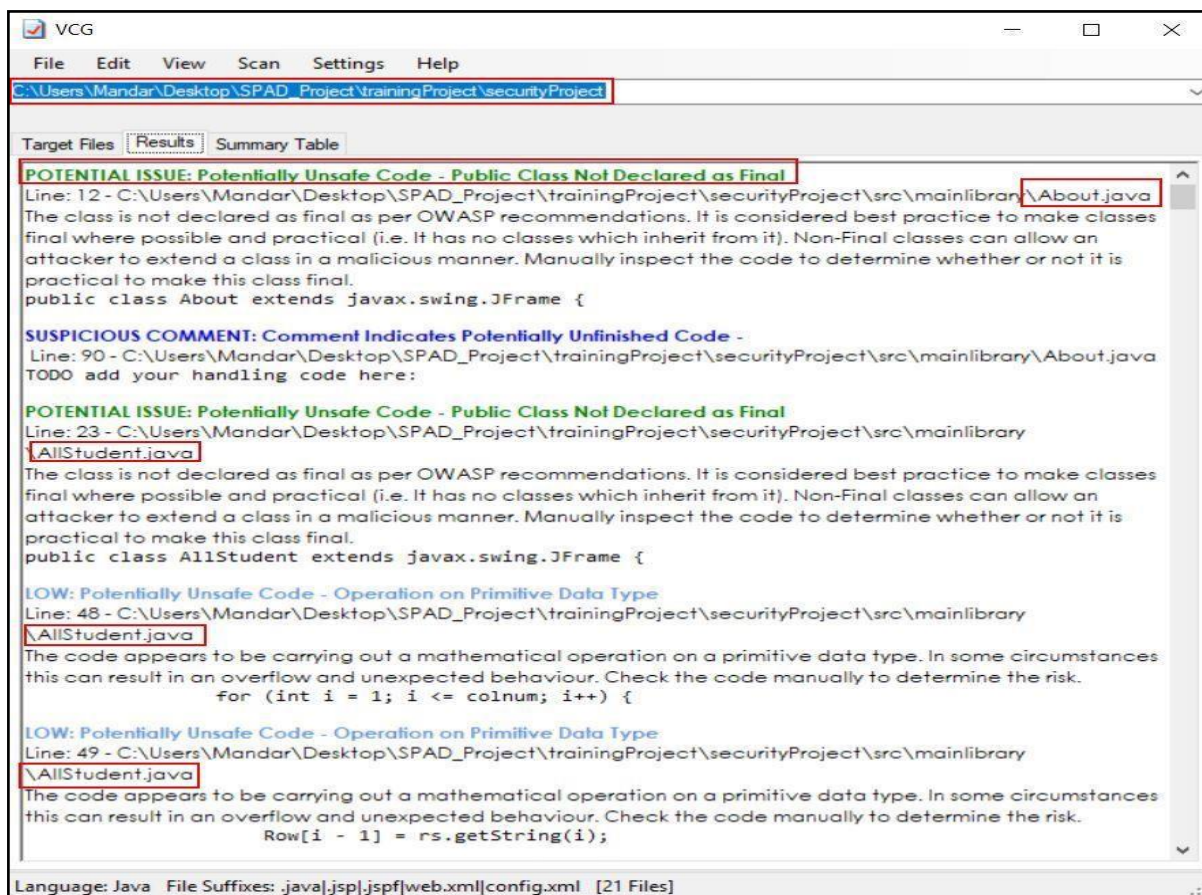
**LOW: Potentially Unsafe Code - Operation on Primitive Data Type**  
 Line: 264 - C:\Users\Mandar\Desktop\SPAD\_Project\trainingProject\securityProject\src\mainlibrary  
 \AllStudent.java  
 The code appears to be carrying out a mathematical operation on a primitive data type. In some circumstances this can result in an overflow and unexpected behaviour. Check the code manually to determine the risk.  
 for (int i = 1; i <= colnum; i++) {

**LOW: Potentially Unsafe Code - Operation on Primitive Data Type**  
 Line: 265 - C:\Users\Mandar\Desktop\SPAD\_Project\trainingProject\securityProject\src\mainlibrary  
 \AllStudent.java  
 The code appears to be carrying out a mathematical operation on a primitive data type. In some circumstances this can result in an overflow and unexpected behaviour. Check the code manually to determine the risk.  
 Row[i - 1] = rs.getString(i);

**LOW: Potentially Unsafe Code - Operation on Primitive Data Type**

Language: Java File Suffixes: .java|.jsp|.jspx|web.xml|config.xml [21 Files]





## V. Proposed Solutions: -

1. One of the main problems we find in this program is that it is vulnerable to an attack by SQL injection. We need to do input validation and use parameterized statements to resolve sql injection issues. To this end, we use Regex features for alphanumeric characters which do not require any special characters to be executed in the field of data.
2. The Second vulnerability we found in the code was the hard coding of SQL credentials in the DB.java. So we've removed passwords from the database and transferred them to another file.
3. Error handling: The program involves loads of "printStackTrace" commands without proper sanitization. That could expose the attacker to information. Error management can be done for personalised error pages.
4. Input data validation: You will verify the user input data such as Username, Password, Book names, Date Ranges, Search queries. Relevant constraints on input data restrictions will be enforced.
5. Code Signing: When completely developed or before publication, the code should be signed with the certificate from the developer. This guarantees system consistency, and the user will detect the same when the system is tampered with.
6. Password Complexity: If the password is not complex enough then attacker will use automated tools to execute a password guessing attack and brute force attack. Password should be at least 8 characters long, 1 digit, 1 small alphabetical character, 1 special character and 1 capital letter.
7. Event Logging: In case of an accident or event, logging is a critical part of the incident management. Application should produce appropriate logs for every occurrence occurring during its execution.
8. Password Hashing: It has been found that the program stores user credentials in the database in cleartext format. Before you transfer to the achieve, the account credentials will be hashed.

## VI. Conclusion: -

For this project, we have performed static analysis client application and reported the result. Nevertheless, dynamic code analysis is also needed to look for more nuanced bugs. Because the framework was two-tier thick client, it was impossible to do dynamic analysis of the source code. But we may still do dynamic analysis with the aid of test cases provided by OWAPS. Along with these we also provided various approaches to security testing which were discussed in their work. We cannot depend solely on such methods, however, because they do not have enough proof that it can capture all the complicated security in application. Thus, more precise security model for application security testing is required. While there is one present model called DIAMOND, which is the fusion of two features called risk-based protection testing and feature-based fuzzing, this feature is currently being tested. Through this project I studied the White box testing methodology, the Java code analysis and the automation technique using Visual Code Grepper (VCG) for security analysis.

## VII. Reference:

- [1] [1]"Programming paradigm", *En.wikipedia.org*, 2020. [Online]. Available: [https://en.wikipedia.org/wiki/Programming\\_paradigm](https://en.wikipedia.org/wiki/Programming_paradigm). [Accessed: 26- Apr- 2020]
- [2] [2]S. Senger de Souza, M. Paiva Prado, E. Francine Barbosa and J. Maldonado, "An Experimental Study to Evaluate the Impact of the Programming Paradigm in the Testing Activity", *CLEI Electronic Journal*, vol. 15, no. 1, 2012.
- [3] *Principles of Programming Languages*, Ben-Gurion University of the Negev: Department of Computer Science, 2017.
- [4] G. Booch, *Object-oriented analysis and design with applications* (2nd ed.), Rational, Santa Clara, CA: Benjamin-Cummings Publishing Co, 1993.
- [5] M. T. A. Ahmad, "A measurement based comparative evaluation of effectiveness of object-oriented versus conventional procedural programming techniques and languages," in *IEEE*, Gold Coast, Queensland, Australia, 2002.
- [6] R. B. Luis Moura e Silva, "Parallel Programming Models," *Semanticscholar*, Melbourne, Australia.
- [7] O. Torgersson, *A Note on Declarative Programming Paradigms*, Sweden.
- [8] A. Shanley and M. N. Johnstone, "Selection of penetration testing methodologies," in *Australian Information Security Management*, Edith Cowan University Australia, 2015.
- [9] S. Jain, R. Johari and A. Kaur, "PJCT: Penetration testing based JAVA code testing tool," in *IEEE*, Noida, India, 2015.
- [10] Q. Ashfaq, R. Khan and S. Farooq, "A Comparative Analysis of Static Code Analysis Tools that check Java Code Adherence to Java Coding Standards," in *IEEE*, Islamabad, Pakistan, , 2019.