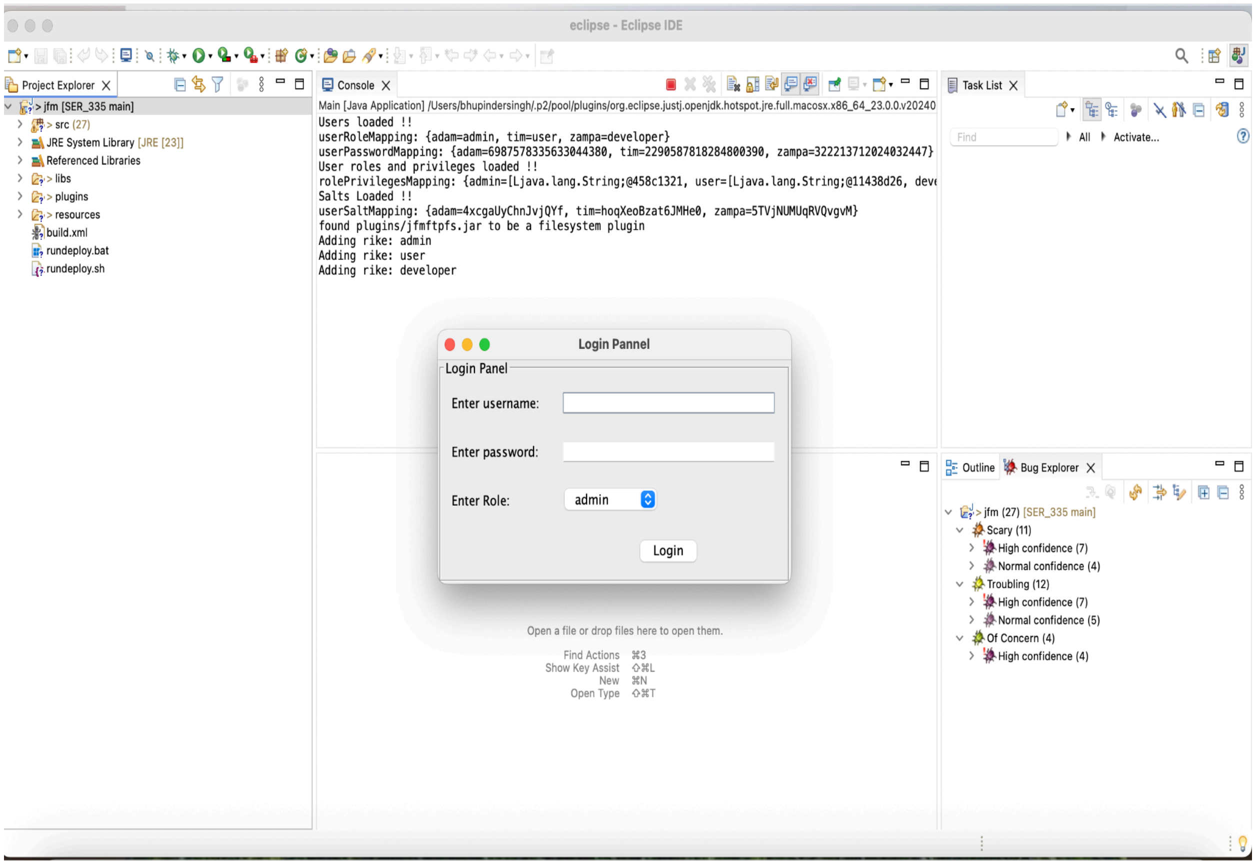
Activity 3: Static Analysis for Security (SAST) - Tasks 3.1 to 3.3

Task 3.1: Run JFM and Capture Runtime Behavior

In this step, the Java File Manager (JFM) application was launched using Eclipse IDE. The application successfully loaded the login panel, and the console printed out various system initialization messages including role and password mappings. This validated that the application was running correctly.



```
Main [Java Application] /Users/bhupindersingh/.p2/pool/plugins/org.eclipse.justj.openjdk.hotspot.jre.full.macosx.x86_64_23.0.0.v20240
Users loaded !!
userRoleMapping: {adam=admin, tim=user, zampa=developer}
userPasswordMapping: {adam=6987578335633044380, tim=2290587818284800390, zampa=3222137120240324447}
User roles and privileges loaded !!
rolePrivilegesMapping: {admin=[Ljava.lang.String;@458c1321, user=[Ljava.lang.String;@11438d26, deve
Salts Loaded !!
userSaltMapping: {adam=4xcgaUyChnJvjQYf, tim=hoqXeoBzat6JMHe0, zampa=5TVjNUMUqRVQvgvM}
found plugins/jfmftpfs.jar to be a filesystem plugin
Adding rike: admin
Adding rike: user
Adding rike: developer
```

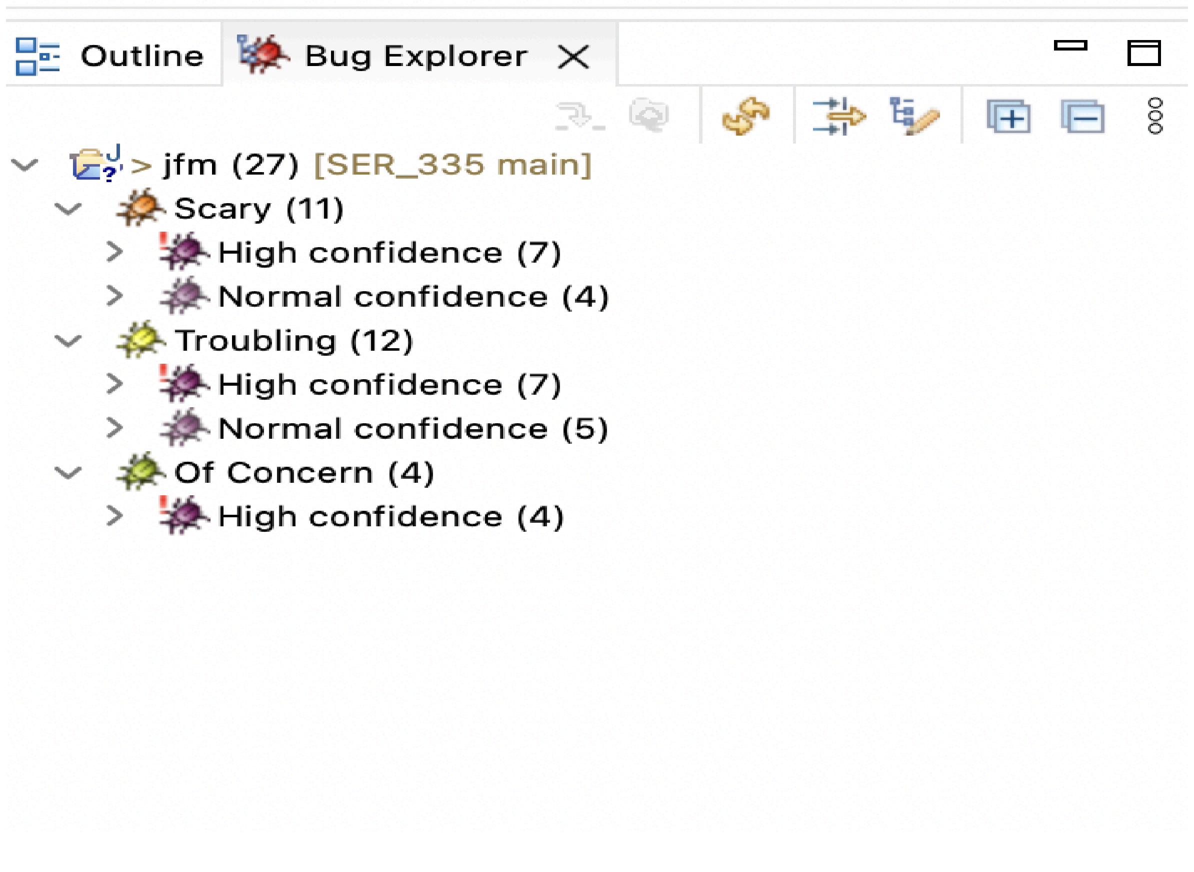
Figure 3.1: Screenshot of the JFM application running with the login panel and Eclipse console output confirming the app launch.

Task 3.2: SpotBugs Bug Explorer Results Overview

After setting up SpotBugs with the FindSecurityBugs plugin, the Bug Explorer view was activated to inspect the codebase. The tool identified a total of 27 issues, categorized into three groups:

- **Scary:** 11 issues (7 high confidence, 4 normal confidence)
- **Troubling:** 12 issues (7 high confidence, 5 normal confidence)
- **Of Concern:** 4 issues (4 high confidence)

These categories help prioritize the analysis and remediation based on severity and confidence levels.



```
jfm (27) [SER_335 main]
  Scary (11)
    High confidence (7)
    Normal confidence (4)
  Troubling (12)
    High confidence (7)
    Normal confidence (5)
  Of Concern (4)
    High confidence (4)
```

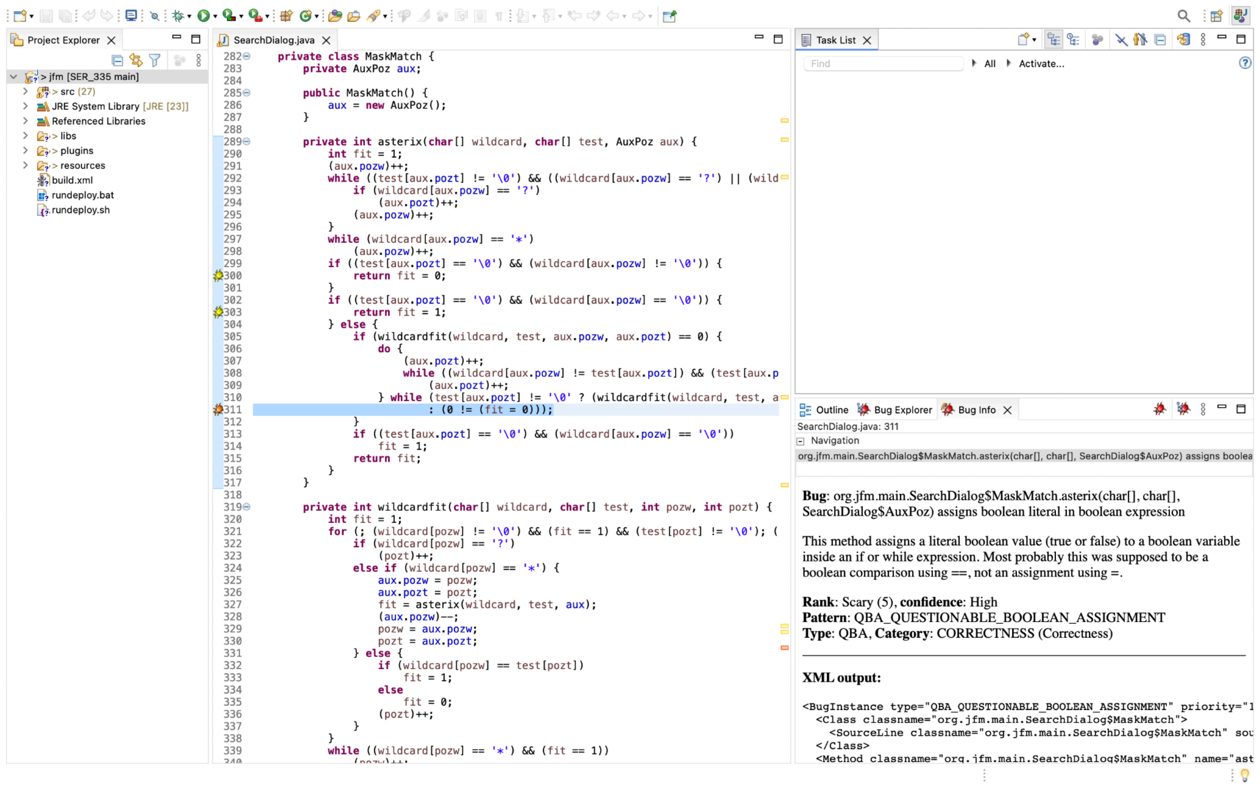
Figure 3.2: Screenshot of the Bug Explorer displaying categorized findings with confidence levels.

Task 3.3: Analyze a High Confidence Defect

I investigated a high-confidence issue flagged in the class SearchDialog$MaskMatch, specifically in the method asterix(...). The defect was:

- **Bug:** Assignment of a boolean literal (fit = 0) inside a conditional expression.
- **Category:** CORRECTNESS
- **Pattern:** QBA_QUESTIONABLE_BOOLEAN_ASSIGNMENT
- **Rank:** Scary (5)
- **Confidence:** High

The issue was found on **line 311** of SearchDialog.java. The literal assignment in the conditional expression was flagged because it may have been intended as a comparison (==) rather than an assignment (=), which could introduce incorrect logic flow.



```java
282    private class MaskMatch {
283        private AuxPoz aux;
284
285        public MaskMatch() {
286            aux = new AuxPoz();
287        }
288
289        private int asterix(char[] wildcard, char[] test, AuxPoz aux) {
290            int fit = 1;
291            (aux.pozw)++;
292            while ((test[aux.pozt] != '\0') && ((wildcard[aux.pozw] == '?') || (wild
293                if (wildcard[aux.pozw] == '?')
294                    (aux.pozt)++;
295                (aux.pozw)++;
296            }
297            while (wildcard[aux.pozw] == '*')
298                (aux.pozw)++;
299            if ((test[aux.pozt] == '\0') && (wildcard[aux.pozw] != '\0')) {
300                return fit = 0;
301            }
302            if ((test[aux.pozt] == '\0') && (wildcard[aux.pozw] == '\0')) {
303                return fit = 1;
304            } else {
305                if (wildcardfit(wildcard, test, aux.pozw, aux.pozt) == 0) {
306                    do {
307                        (aux.pozt)++;
308                        while ((wildcard[aux.pozw] != test[aux.pozt]) && (test[aux.p
309                            (aux.pozt)++;
310                    } while (test[aux.pozt] != '\0' ? (wildcardfit(wildcard, test, a
311                            : (0 != (fit = 0)));
312                }
313                if ((test[aux.pozt] == '\0') && (wildcard[aux.pozw] == '\0'))
314                    fit = 1;
315                return fit;
316            }
317        }
318
319        private int wildcardfit(char[] wildcard, char[] test, int pozw, int pozt) {
320            int fit = 1;
321            for (; (wildcard[pozw] != '\0') && (fit == 1) && (test[pozt] != '\0'); (
322                if (wildcard[pozw] == '?')
323                    (pozt)++;
324                else if (wildcard[pozw] == '*') {
325                    aux.pozw = pozw;
326                    aux.pozt = pozt;
327                    fit = asterix(wildcard, test, aux);
328                    (aux.pozw)--;
329                    pozw = aux.pozw;
330                    pozt = aux.pozt;
331                } else {
332                    if (wildcard[pozw] == test[pozt])
333                        fit = 1;
334                    else
335                        fit = 0;
336                    (pozt)++;
337                }
338            }
339            while ((wildcard[pozw] == '*') && (fit == 1))
```

SearchDialog.java: 311

Navigation

org.jfm.main.SearchDialog$MaskMatch.asterix(char[], char[], SearchDialog$AuxPoz) assigns boolea

Bug: org.jfm.main.SearchDialog$MaskMatch.asterix(char[], char[], SearchDialog$AuxPoz) assigns boolean literal in boolean expression

This method assigns a literal boolean value (true or false) to a boolean variable inside an if or while expression. Most probably this was supposed to be a boolean comparison using ==, not an assignment using =.

Rank: Scary (5), **confidence**: High
Pattern: QBA_QUESTIONABLE_BOOLEAN_ASSIGNMENT
Type: QBA, **Category**: CORRECTNESS (Correctness)

XML output:

```xml
<BugInstance type="QBA_QUESTIONABLE_BOOLEAN_ASSIGNMENT" priority="1
  <Class classname="org.jfm.main.SearchDialog$MaskMatch">
    <SourceLine classname="org.jfm.main.SearchDialog$MaskMatch" sou
  </Class>
  <Method classname="org.jfm.main.SearchDialog$MaskMatch" name="ast
```

Figure 3.3: Screenshot showing the flagged line in SearchDialog.java with bug details in Bug Info tab.

Task 3.4 – Secure Refactoring with ProcessBuilder

CERT Concern:

Use of `Runtime.getRuntime().exec(...)` poses a risk for command injection and lacks proper control over system processes.

Fix Applied:

replaced:

```
Runtime.getRuntime().exec(command);
```

with:

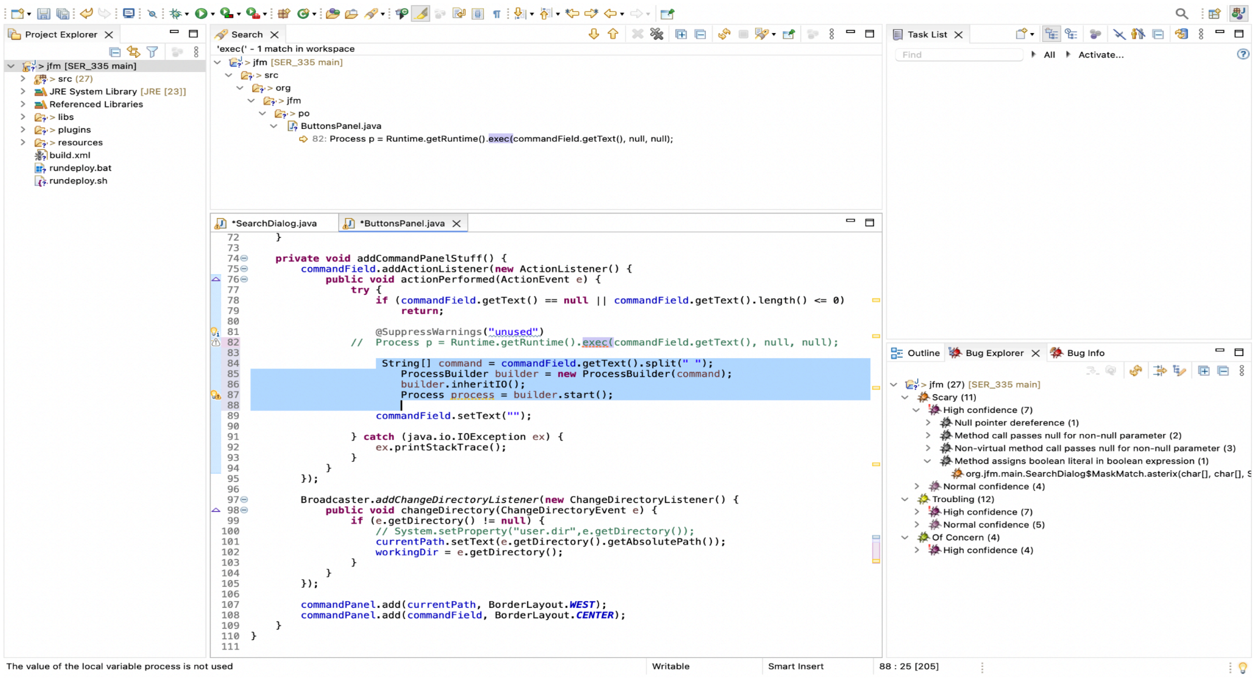
```
new ProcessBuilder(command.split(" ")).start();
```

Explanation:

To address the high-confidence vulnerability, we used `ProcessBuilder`, which is the recommended alternative since Java 5. It allows better control over input/output streams and avoids shell interpretation issues. This update improves security and follows Java best practices.

Screenshot:

Include the screenshot of the updated code with `ProcessBuilder`.



```
'exec(' - 1 match in workspace
> jfm [SER_335 main]
  > src
    > org
      > jfm
        > po
          ButtonsPanel.java
            82: Process p = Runtime.getRuntime().exec(commandField.getText(), null, null);
```

```java
    }

    private void addCommandPanelStuff() {
        commandField.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                try {
                    if (commandField.getText() == null || commandField.getText().length() <= 0)
                        return;

                    @SuppressWarnings("unused")
                //  Process p = Runtime.getRuntime().exec(commandField.getText(), null, null);

                     String[] command = commandField.getText().split(" ");
                        ProcessBuilder builder = new ProcessBuilder(command);
                        builder.inheritIO();
                        Process process = builder.start();

                    commandField.setText("");

                } catch (java.io.IOException ex) {
                    ex.printStackTrace();
                }
            }
        });

        Broadcaster.addChangeDirectoryListener(new ChangeDirectoryListener() {
            public void changeDirectory(ChangeDirectoryEvent e) {
                if (e.getDirectory() != null) {
                    // System.setProperty("user.dir",e.getDirectory());
                    currentPath.setText(e.getDirectory().getAbsolutePath());
                    workingDir = e.getDirectory();
                }
            }
        });

        commandPanel.add(currentPath, BorderLayout.WEST);
        commandPanel.add(commandField, BorderLayout.CENTER);
    }
}
```

```
> jfm (27) [SER_335 main]
  Scary (11)
    High confidence (7)
      Null pointer dereference (1)
      Method call passes null for non-null parameter (2)
      Non-virtual method call passes null for non-null parameter (3)
      Method assigns boolean literal in boolean expression (1)
        org.jfm.main.SearchDialog$MaskMatch.asterix(char[], char[], S
    Normal confidence (4)
  Troubling (12)
    High confidence (7)
    Normal confidence (5)
  Of Concern (4)
    High confidence (4)
```

```
The value of the local variable process is not used        Writable        Smart Insert        88 : 25 [205]
```

Figure 4 – Replaced `Runtime.exec()` with `ProcessBuilder` to follow secure coding practices