Testing Analysis

SWE303

Worked: Evisa Nela

Accepted by: Ari Gjerazi, Jurgen Cama

This testing analysis covers three methods from the POS system using suitable testing techniques. Boundary Value Testing was used for the isSessionExpired() method to check behavior around the session expiration limit. Decision Table / Equivalence Class Testing was applied to the isSuccessfull() method, since its result depends on message types. The getErrorMessages() method was tested to ensure correct filtering and formatting of error and exception messages. All tests were implemented as unit tests using JUnit 5.

Method 1: isSessionExpired()

```
19      public boolean isSessionExpired() {  7 usages
20          if (sessionStartTime == null) {
21              return true;
22          }
23          long currentTimeMillis = System.currentTimeMillis();
24          long sessionDurationMillis = 15 * 60 * 1000; // 15 minutes in milliseconds
25          long sessionEndTimeMillis = sessionStartTime.getTime() + sessionDurationMillis;
26
27          return currentTimeMillis > sessionEndTimeMillis;
28      }
```

Package and class:

- **Package:** model
- **Class:** ApplicationSession.java

Purpose:

Checks whether the current application session has expired.
The session is considered expired when:

- sessionStartTime is **null** (session not started), or
- the current time is **greater than** the session end time (15 minutes after start).

Test Design Technique:

Boundary Value Testing (BVT)

Session is considered valid when:

sessionStartTime != null AND elapsed time **≤ 15 minutes**

Session is considered expired when:

sessionStartTime == null OR elapsed time **> 15 minutes**

Boundary:

The critical boundary is at **15 minutes**.

⚠ Important detail (based on your code):
Your code uses > not >=, so:

- **exactly 15:00 is NOT expired**
- **after 15:00 (15:00 + 1ms or 15:01) IS expired**

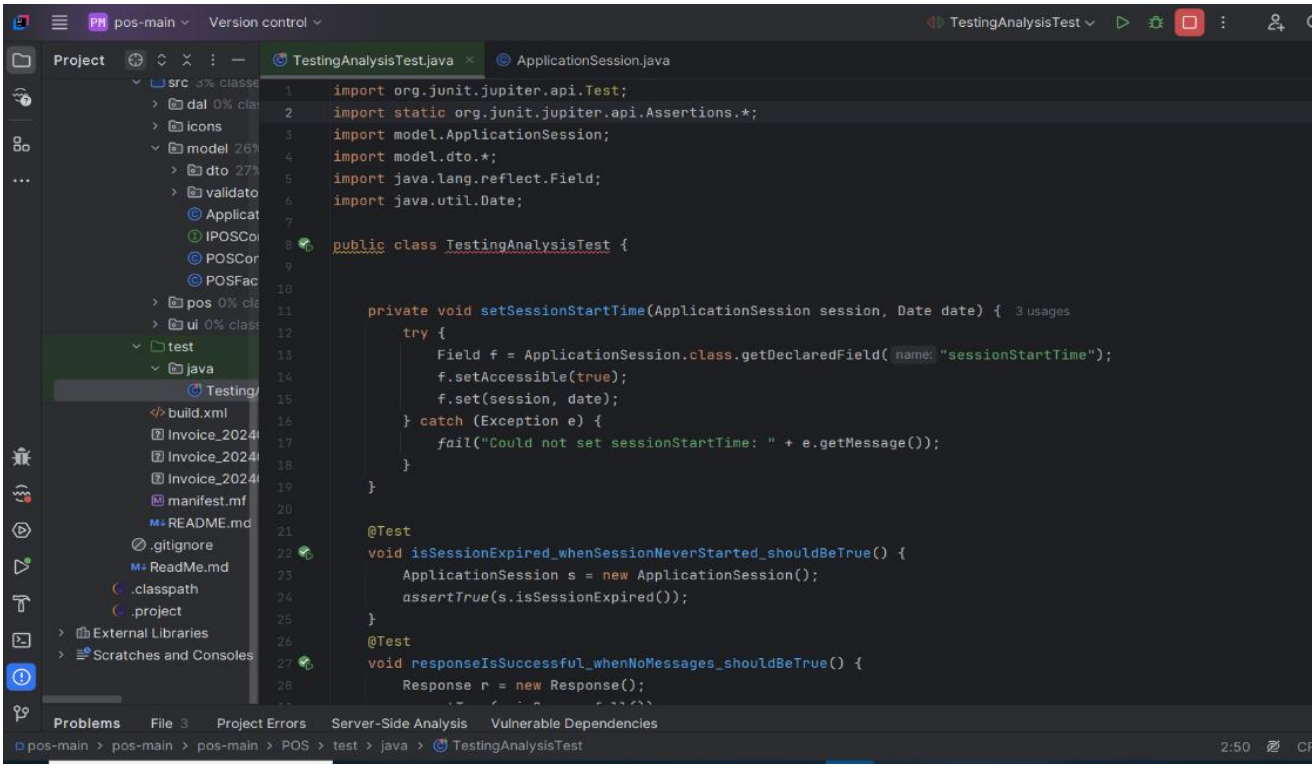
Boundary	Session Start Time Example	Elapsed Time	Expected Behavior
Min−1	startTime set	14:59	Not expired (false)
Min	startTime set	15:00	Not expired (false)
Min+1	startTime set	15:01 (or 15:00 + 1ms)	Expired (true)
Nominal valid	startTime set	05:00	Not expired (false)
Special case	startTime = null	—	Expired (true)

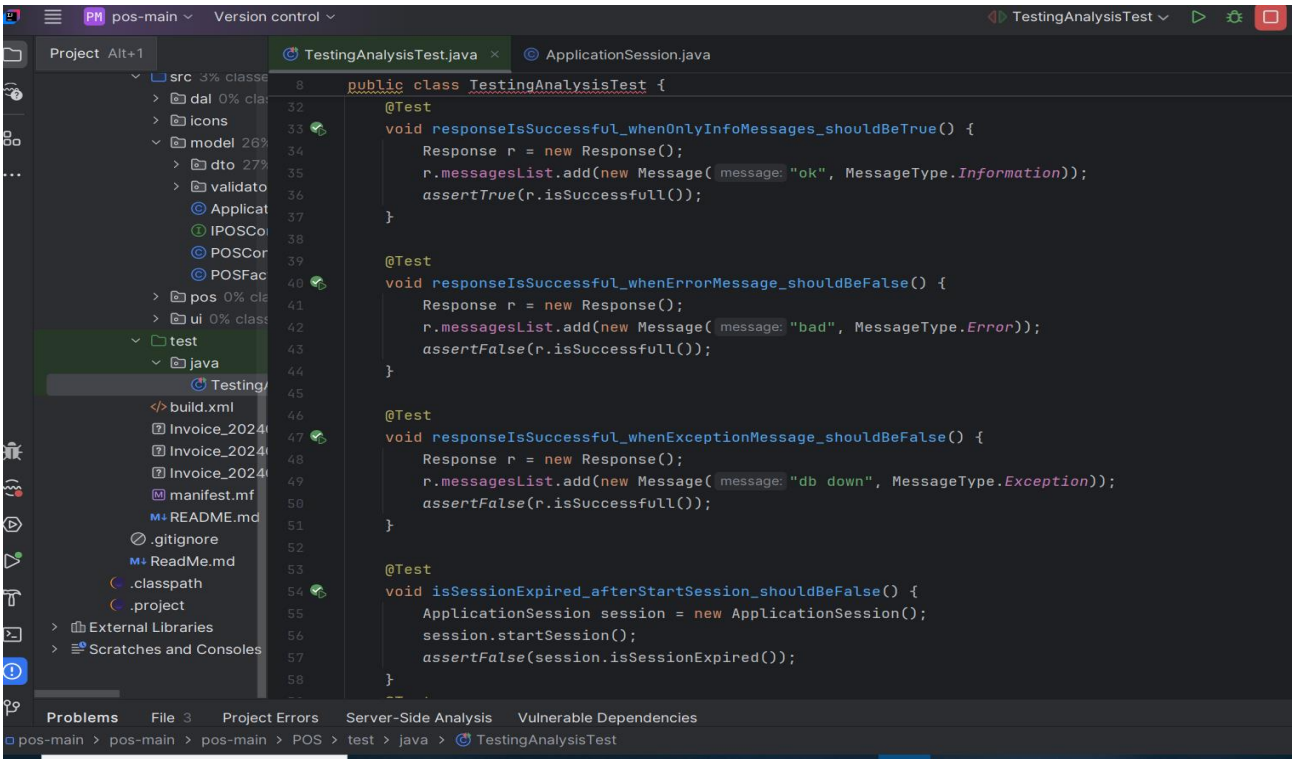
Preconditions (for all cases):

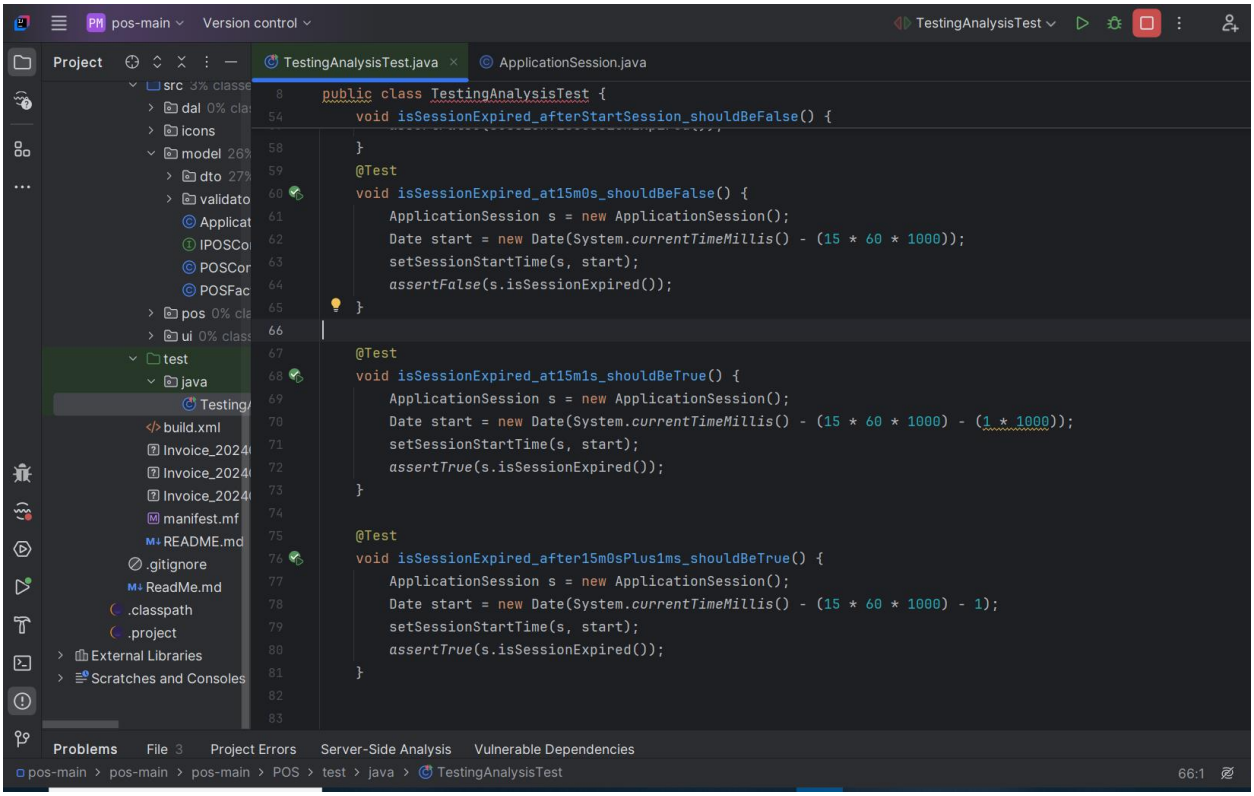
- ApplicationSession object is created successfully.
- System time is available (System.currentTimeMillis() works normally).
- sessionStartTime is either:
  - left null, or
  - set to a valid Date before calling isSessionExpired().

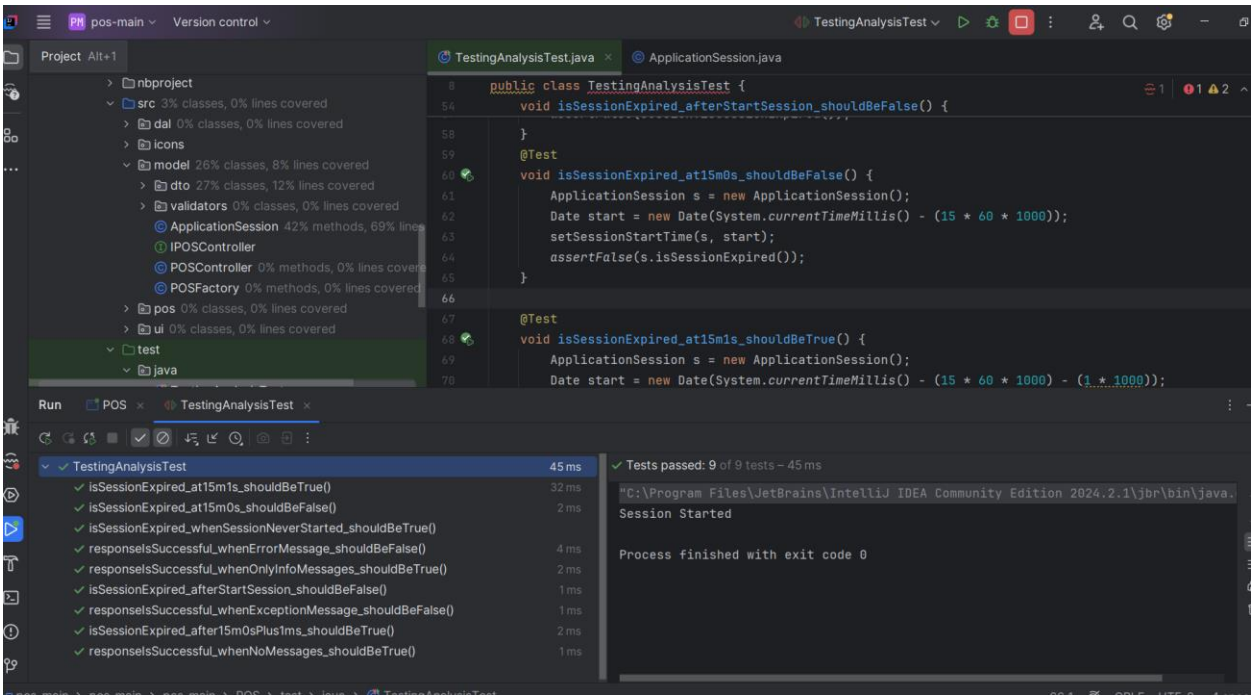
Test Case Table

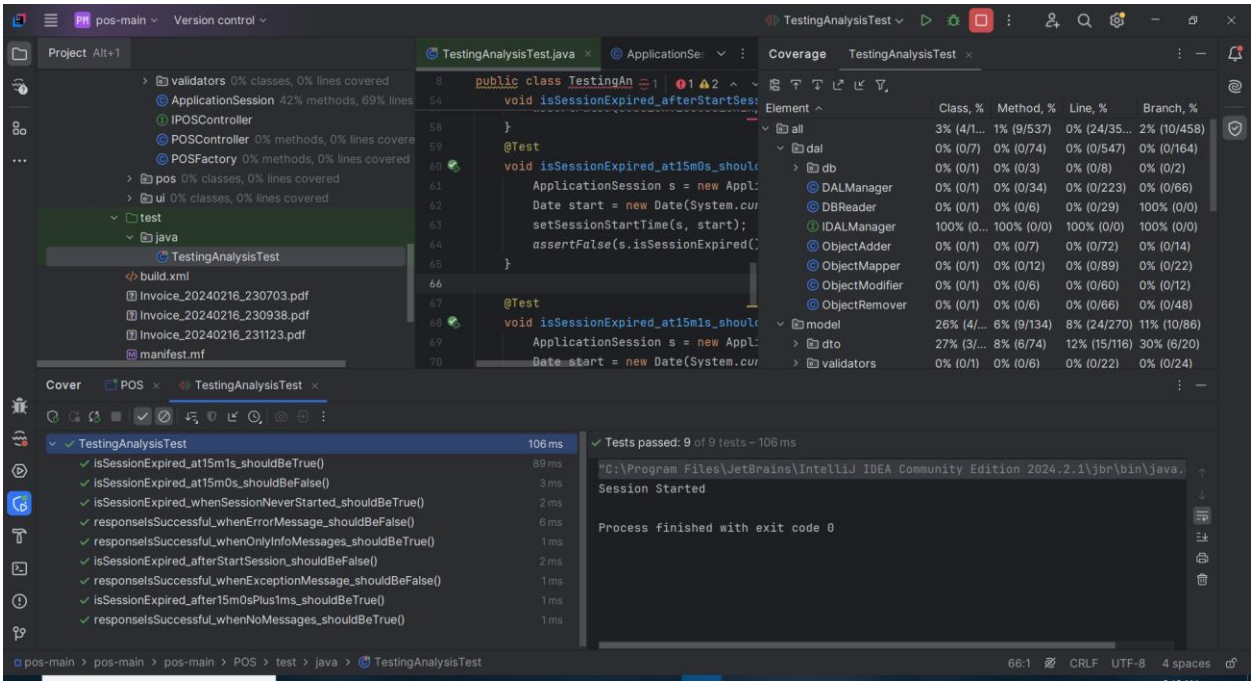
Test Case ID	Input Condition	Expected Result
TC-SES-001	sessionStartTime = null	returns true
TC-SES-002	call startSession() then call isSessionExpired()	returns false
TC-SES-003	startTime = now − 14m59s	returns false
TC-SES-004	startTime = now − 15m00s	returns false
TC-SES-005	startTime = now − 15m01s	returns true


pos-main
Version control
TestingAnalysisTest
Project
TestingAnalysisTest.java
ApplicationSession.java
import org.junit.jupiter.api.Test;
import static org.junit.jupiter.api.Assertions.*;
import model.ApplicationSession;
import model.dto.*;
import java.lang.reflect.Field;
import java.util.Date;
public class TestingAnalysisTest {
private void setSessionStartTime(ApplicationSession session, Date date) { 3 usages
try {
Field f = ApplicationSession.class.getDeclaredField(name: "sessionStartTime");
f.setAccessible(true);
f.set(session, date);
} catch (Exception e) {
fail("Could not set sessionStartTime: " + e.getMessage());
}
}
@Test
void isSessionExpired_whenSessionNeverStarted_shouldBeTrue() {
ApplicationSession s = new ApplicationSession();
assertTrue(s.isSessionExpired());
}
@Test
void responseIsSuccessful_whenNoMessages_shouldBeTrue() {
Response r = new Response();
Problems
File 3
Project Errors
Server-Side Analysis
Vulnerable Dependencies
pos-main > pos-main > pos-main > POS > test > java > TestingAnalysisTest
2:50

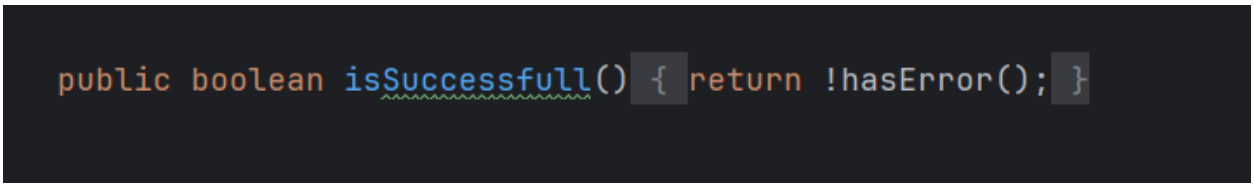

pos-main
Version control
TestingAnalysisTest
Project Alt+1
TestingAnalysisTest.java
ApplicationSession.java
public class TestingAnalysisTest {
@Test
void responseIsSuccessful_whenOnlyInfoMessages_shouldBeTrue() {
Response r = new Response();
r.messagesList.add(new Message(message: "ok", MessageType.Information));
assertTrue(r.isSuccessfull());
}
@Test
void responseIsSuccessful_whenErrorMessage_shouldBeFalse() {
Response r = new Response();
r.messagesList.add(new Message(message: "bad", MessageType.Error));
assertFalse(r.isSuccessfull());
}
@Test
void responseIsSuccessful_whenExceptionMessage_shouldBeFalse() {
Response r = new Response();
r.messagesList.add(new Message(message: "db down", MessageType.Exception));
assertFalse(r.isSuccessfull());
}
@Test
void isSessionExpired_afterStartSession_shouldBeFalse() {
ApplicationSession session = new ApplicationSession();
session.startSession();
assertFalse(session.isSessionExpired());
}
Problems
File 3
Project Errors
Server-Side Analysis
Vulnerable Dependencies
pos-main > pos-main > pos-main > POS > test > java > TestingAnalysisTest


pos-main
Version control
TestingAnalysisTest
Project
TestingAnalysisTest.java
ApplicationSession.java
public class TestingAnalysisTest {
void isSessionExpired_afterStartSession_shouldBeFalse() {
}
@Test
void isSessionExpired_at15m0s_shouldBeFalse() {
ApplicationSession s = new ApplicationSession();
Date start = new Date(System.currentTimeMillis() - (15 * 60 * 1000));
setSessionStartTime(s, start);
assertFalse(s.isSessionExpired());
}
@Test
void isSessionExpired_at15m1s_shouldBeTrue() {
ApplicationSession s = new ApplicationSession();
Date start = new Date(System.currentTimeMillis() - (15 * 60 * 1000) - (1 * 1000));
setSessionStartTime(s, start);
assertTrue(s.isSessionExpired());
}
@Test
void isSessionExpired_after15m0sPlus1ms_shouldBeTrue() {
ApplicationSession s = new ApplicationSession();
Date start = new Date(System.currentTimeMillis() - (15 * 60 * 1000) - 1);
setSessionStartTime(s, start);
assertTrue(s.isSessionExpired());
}
Problems
File 3
Project Errors
Server-Side Analysis
Vulnerable Dependencies
66:1


pos-main
Version control
TestingAnalysisTest
Project Alt+1
TestingAnalysisTest.java
ApplicationSession.java
Run
POS
TestingAnalysisTest
TestingAnalysisTest 45 ms
isSessionExpired_at15m1s_shouldBeTrue() 32 ms
isSessionExpired_at15m0s_shouldBeFalse() 2 ms
isSessionExpired_whenSessionNeverStarted_shouldBeTrue()
responseIsSuccessful_whenErrorMessage_shouldBeFalse() 4 ms
responseIsSuccessful_whenOnlyInfoMessages_shouldBeTrue() 2 ms
isSessionExpired_afterStartSession_shouldBeFalse() 1 ms
responseIsSuccessful_whenExceptionMessage_shouldBeFalse() 1 ms
isSessionExpired_after15m0sPlus1ms_shouldBeTrue() 2 ms
responseIsSuccessful_whenNoMessages_shouldBeTrue() 1 ms
Tests passed: 9 of 9 tests – 45 ms
"C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2024.2.1\jbr\bin\java.
Session Started
Process finished with exit code 0



Method 2: Response.isSuccessfull()




```
public boolean isSuccessfull() {
   return !hasError();
}
```

and hasError():

```
private boolean hasError() {
   for(Message m : messagesList) {
      if(m.type == MessageType.Error || m.type == MessageType.Exception)
         return true;
   }
   return false;
}
```

```java
    private boolean hasError() {  1 usage
        for(Message m : messagesList)
        {
            if(m.type == MessageType.Error || m.type == MessageType.Exception)
                return true;
        }
        return false;
}
```

- If there is **any** Error or Exception message → **isSuccessfull() = false**
- If there are **no** error/exception messages (even if Info/Warning exists) → **true**

Equivalence Class Testing (ECT)

Boundary	MessagesList Example	Expected Behavior
Min−1	Only Information/Warning messages	**Success = true**
Min	No messages (empty list)	**Success = true**
Min+1	Exactly 1 Error message	**Success = false**
Nominal valid	Multiple Info/Warning only	**Success = true**
Special case	Exactly 1 Exception message	**Success = false**

This matches the idea of Min−1/Min/Min+1 but adapted to list-content.

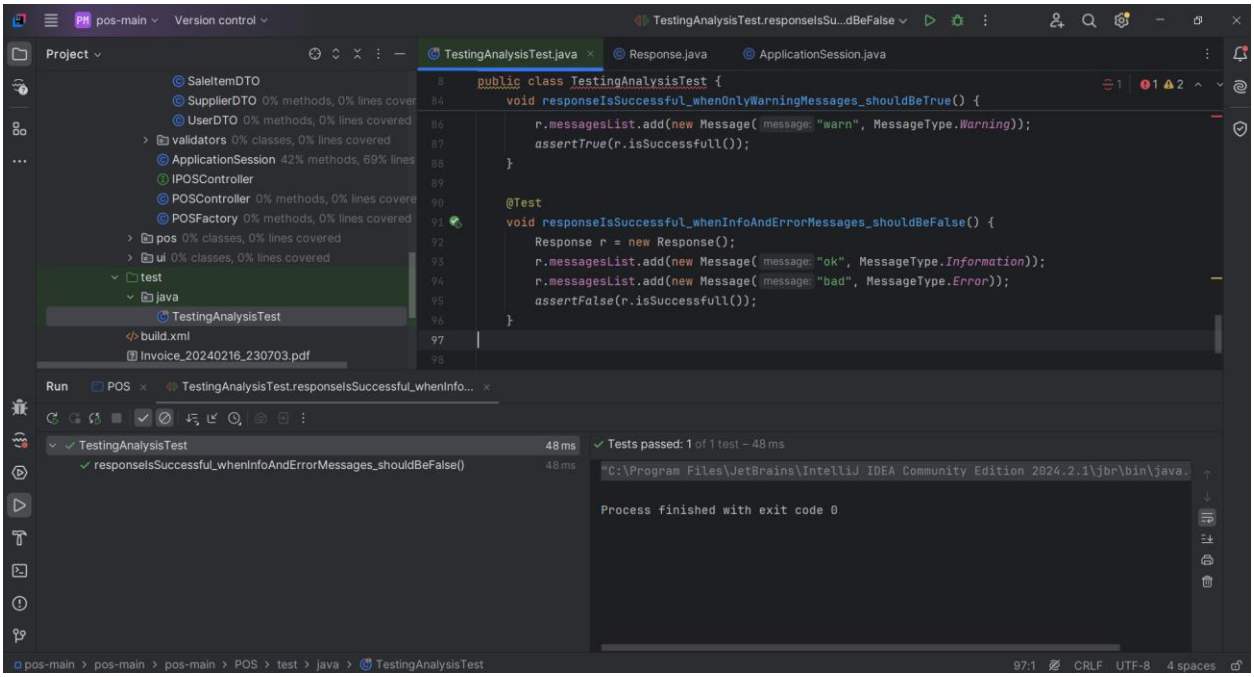
Preconditions

Preconditions (for all cases):

- Response object is created successfully
- messagesList is initialized (constructor creates an empty ArrayList)

Test Case Table

Test Case ID	Input messages	Expected Result
TC-RES-001	empty messagesList	isSuccessfull() returns **true**
TC-RES-002	add Information message	returns **true**
TC-RES-003	add Warning message	returns **true**
TC-RES-004	add Error message	returns **false**
TC-RES-005	add Exception message	returns **false**
TC-RES-006	Info + Error together	returns **false**



Method 3: Response.getErrorMessages()

```
public String getErrorMessages() {  8 usages
    StringBuilder sb = new StringBuilder();
    for(Message m : messagesList)
    {
       if(sb.length() > 0)
           sb.append(",\n");
        if(m.type == MessageType.Error || m.type == MessageType.Exception)
            sb.append(m.message);
    }
    return sb.toString();
}
```

1) What the method does

- It loops through messagesList
- It returns a single string containing only **Error** and **Exception** messages
- It separates multiple messages by ",\n"
- If no error/exception exists → it returns an **empty string**

A) Boundary Value Testing (BVT)

The boundary here is **number of Error/Exception messages included** and whether separators are added.

Boundary Table

Boundary	MessagesList Example	Expected Behavior
Min−1	Only Information/Warning messages	returns empty string ""
Min	Exactly 1 Error message	returns "errorText" (no comma/newline)
Min+1	2 error-type messages	returns "m1,\n m2" (separator appears)
Nominal valid	Many errors/exceptions mixed with info	returns only errors/exceptions joined
Special case	messagesList is empty	returns ""

B) Preconditions

Preconditions (for all cases):

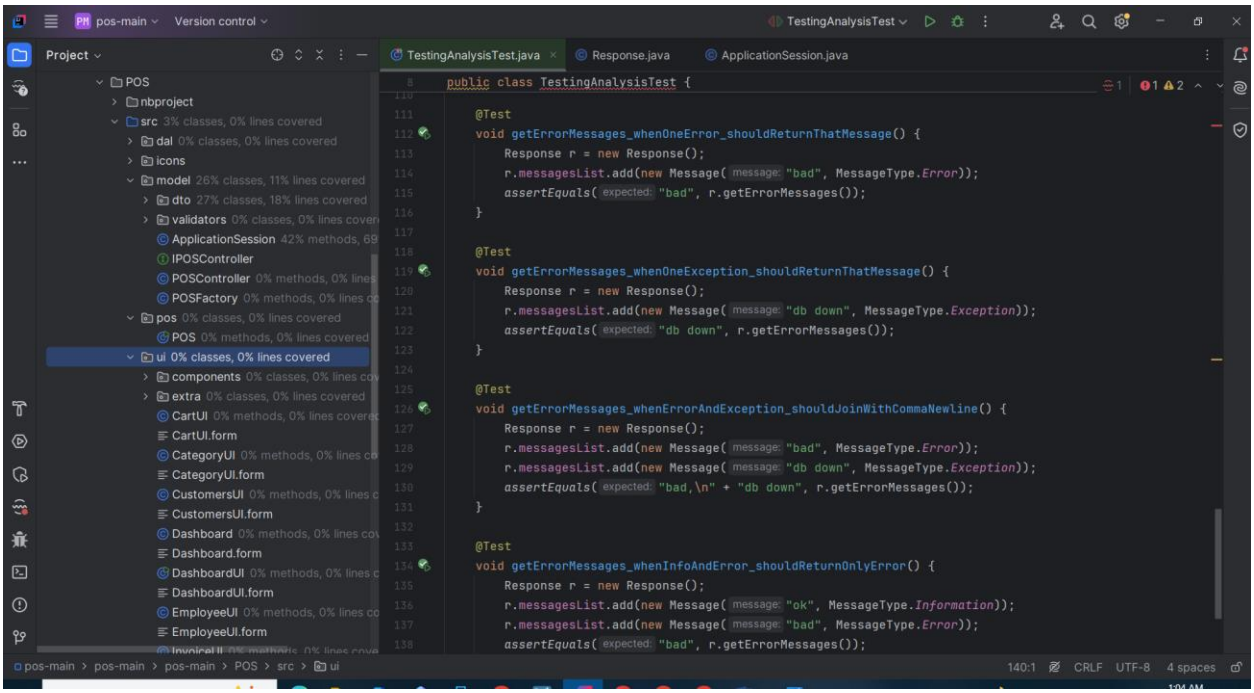
- Response object is created successfully
- messagesList is initialized (constructor creates it)
- Each Message has a non-null message string (use simple text in tests)

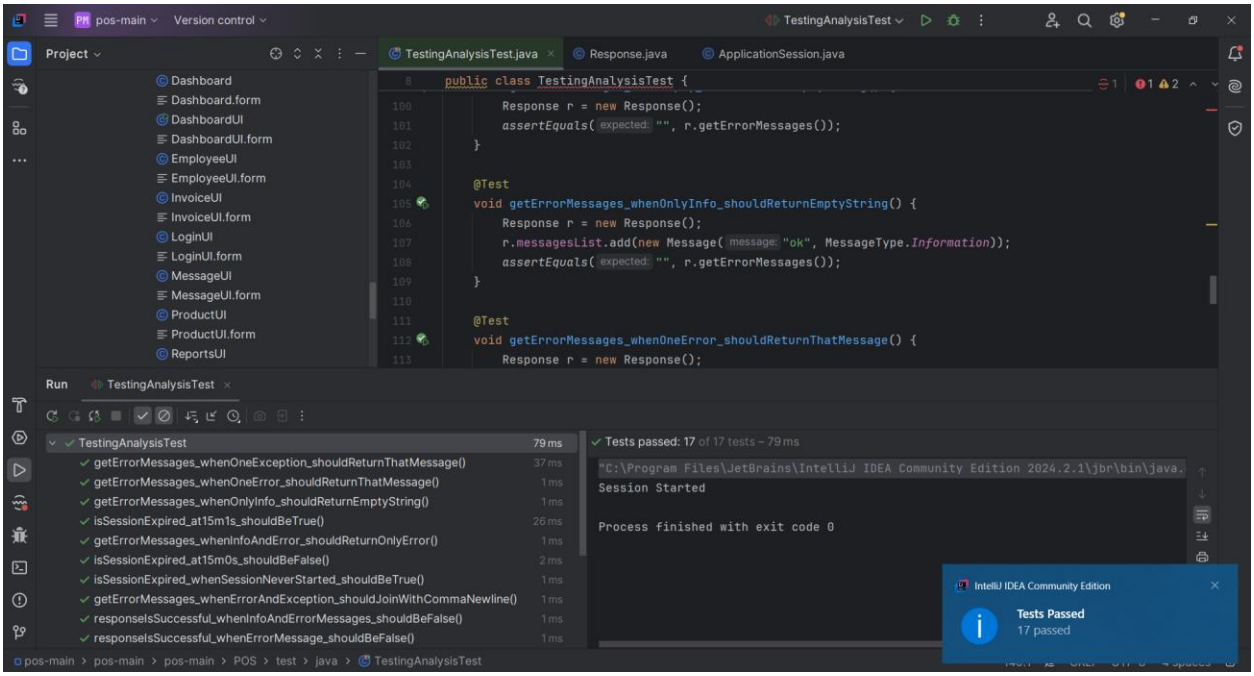
C) Test Case Table

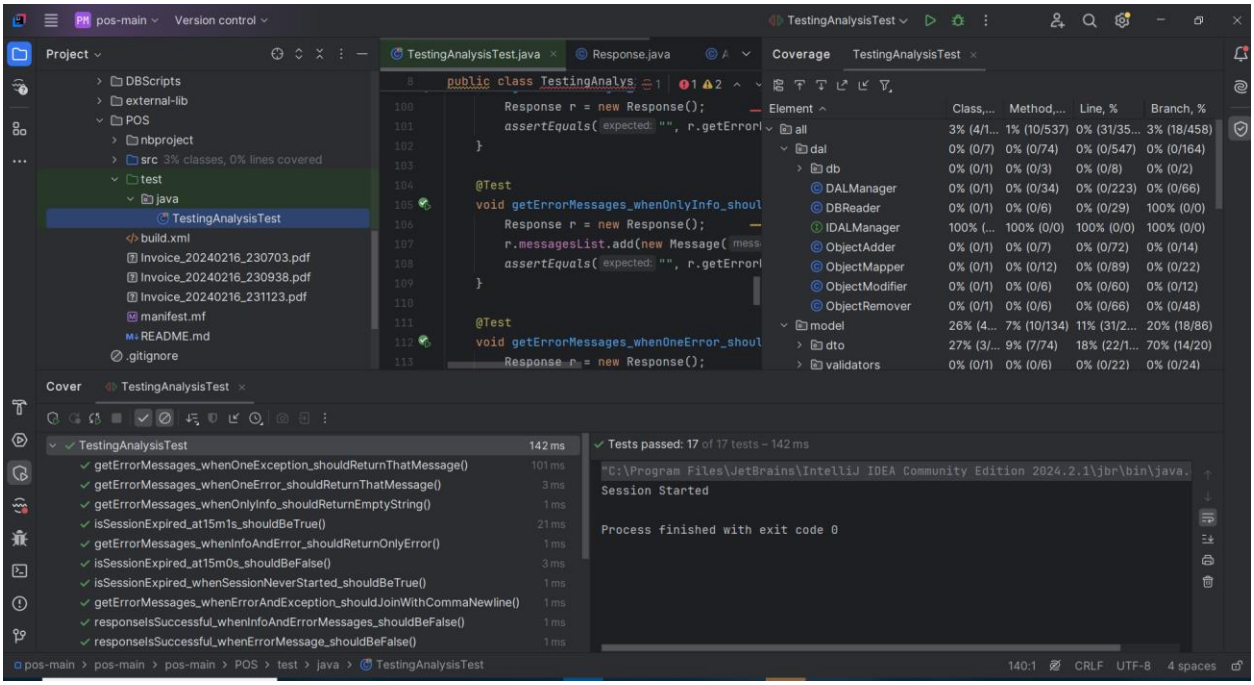
Test Case ID	Input messages	Expected Result
TC-ERR-001	empty list	returns ""
TC-ERR-002	one Information	returns ""
TC-ERR-003	one Error: "bad"	returns "bad"
TC-ERR-004	one Exception: "db down"	returns "db down"
TC-ERR-005	Error "bad" + Exception "db down"	returns "bad,\n db down"
TC-ERR-006	Info "ok" + Error "bad"	returns "bad"

```
@Test
void getErrorMessages_whenEmpty_shouldReturnEmptyString() {
   Response r = new Response();
   assertEquals("", r.getErrorMessages());
}

@Test
void getErrorMessages_whenOnlyInfo_shouldReturnEmptyString() {
   Response r = new Response();
   r.messagesList.add(new Message("ok", MessageType.Information));
   assertEquals("", r.getErrorMessages());
}

@Test
void getErrorMessages_whenOneError_shouldReturnThatMessage() {
   Response r = new Response();
   r.messagesList.add(new Message("bad", MessageType.Error));
   assertEquals("bad", r.getErrorMessages());
}
```

```
@Test
void getErrorMessages_whenOneException_shouldReturnThatMessage() {
   Response r = new Response();
   r.messagesList.add(new Message("db down", MessageType.Exception));
   assertEquals("db down", r.getErrorMessages());
}

@Test
void getErrorMessages_whenErrorAndException_shouldJoinWithCommaNewline() {
   Response r = new Response();
   r.messagesList.add(new Message("bad", MessageType.Error));
   r.messagesList.add(new Message("db down", MessageType.Exception));
   assertEquals("bad,\n" + "db down", r.getErrorMessages());
}

@Test
void getErrorMessages_whenInfoAndError_shouldReturnOnlyError() {
   Response r = new Response();
   r.messagesList.add(new Message("ok", MessageType.Information));
   r.messagesList.add(new Message("bad", MessageType.Error));
   assertEquals("bad", r.getErrorMessages());
}
```




pos-main
Version control
TestingAnalysisTest
Project
TestingAnalysisTest.java
Response.java
ApplicationSession.java
Dashboard
Dashboard.form
DashboardUI
DashboardUI.form
EmployeeUI
EmployeeUI.form
InvoiceUI
InvoiceUI.form
LoginUI
LoginUI.form
MessageUI
MessageUI.form
ProductUI
ProductUI.form
ReportsUI
public class TestingAnalysisTest {
Response r = new Response();
assertEquals(expected: "", r.getErrorMessages());
@Test
void getErrorMessages_whenOnlyInfo_shouldReturnEmptyString() {
Response r = new Response();
r.messagesList.add(new Message(message: "ok", MessageType.Information));
assertEquals(expected: "", r.getErrorMessages());
@Test
void getErrorMessages_whenOneError_shouldReturnThatMessage() {
Response r = new Response();
Run
TestingAnalysisTest
TestingAnalysisTest 79 ms
getErrorMessages_whenOneException_shouldReturnThatMessage() 37 ms
getErrorMessages_whenOneError_shouldReturnThatMessage() 1 ms
getErrorMessages_whenOnlyInfo_shouldReturnEmptyString() 1 ms
isSessionExpired_at15m1s_shouldBeTrue() 26 ms
getErrorMessages_whenInfoAndError_shouldReturnOnlyError() 1 ms
isSessionExpired_at15m0s_shouldBeFalse() 2 ms
isSessionExpired_whenSessionNeverStarted_shouldBeTrue() 1 ms
getErrorMessages_whenErrorAndException_shouldJoinWithCommaNewline() 1 ms
responseIsSuccessful_whenInfoAndErrorMessages_shouldBeFalse() 1 ms
responseIsSuccessful_whenErrorMessage_shouldBeFalse() 1 ms
Tests passed: 17 of 17 tests – 79 ms
"C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2024.2.1\jbr\bin\java.
Session Started
Process finished with exit code 0
IntelliJ IDEA Community Edition
Tests Passed
17 passed
pos-main › pos-main › pos-main › POS › test › java › TestingAnalysisTest


pos-main
Version control
TestingAnalysisTest
Project
DBScripts
external-lib
POS
nbproject
src 3% classes, 0% lines covered
test
java
TestingAnalysisTest
build.xml
Invoice_20240216_230703.pdf
Invoice_20240216_230938.pdf
Invoice_20240216_231123.pdf
manifest.mf
README.md
.gitignore
TestingAnalysisTest.java
Response.java
Coverage
TestingAnalysisTest
Element
Class,...
Method,...
Line, %
Branch, %
all 3% (4/1... 1% (10/537) 0% (31/35... 3% (18/458)
dal 0% (0/7) 0% (0/74) 0% (0/547) 0% (0/164)
db 0% (0/1) 0% (0/3) 0% (0/8) 0% (0/2)
DALManager 0% (0/1) 0% (0/34) 0% (0/223) 0% (0/66)
DBReader 0% (0/1) 0% (0/6) 0% (0/29) 100% (0/0)
IDALManager 100% (... 100% (0/0) 100% (0/0) 100% (0/0)
ObjectAdder 0% (0/1) 0% (0/7) 0% (0/72) 0% (0/14)
ObjectMapper 0% (0/1) 0% (0/12) 0% (0/89) 0% (0/22)
ObjectModifier 0% (0/1) 0% (0/6) 0% (0/60) 0% (0/12)
ObjectRemover 0% (0/1) 0% (0/6) 0% (0/66) 0% (0/48)
model 26% (4... 7% (10/134) 11% (31/2... 20% (18/86)
dto 27% (3/... 9% (7/74) 18% (22/1... 70% (14/20)
validators 0% (0/1) 0% (0/6) 0% (0/22) 0% (0/24)
Cover
TestingAnalysisTest
TestingAnalysisTest 142 ms
getErrorMessages_whenOneException_shouldReturnThatMessage() 101 ms
getErrorMessages_whenOneError_shouldReturnThatMessage() 3 ms
getErrorMessages_whenOnlyInfo_shouldReturnEmptyString() 1 ms
isSessionExpired_at15m1s_shouldBeTrue() 21 ms
getErrorMessages_whenInfoAndError_shouldReturnOnlyError() 1 ms
isSessionExpired_at15m0s_shouldBeFalse() 3 ms
isSessionExpired_whenSessionNeverStarted_shouldBeTrue() 1 ms
getErrorMessages_whenErrorAndException_shouldJoinWithCommaNewline() 1 ms
responseIsSuccessful_whenInfoAndErrorMessages_shouldBeFalse() 1 ms
responseIsSuccessful_whenErrorMessage_shouldBeFalse() 1 ms
Tests passed: 17 of 17 tests – 142 ms
"C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2024.2.1\jbr\bin\java.
Session Started
Process finished with exit code 0
pos-main › pos-main › pos-main › POS › test › java › TestingAnalysisTest
140:1 CRLF UTF-8 4 spaces