# CSCD01 Team 21
# Deliverable #2

Team 21: *The BugWriters*

Chengrong Zhang, Zhongyang Xia, Tan Jiaxin, Zongye Yang,

Xingyuan Zhu

March 2020

# Contents

## 0.1 Project Management

This section shows how Team 21 manages the project in deliverable 2.

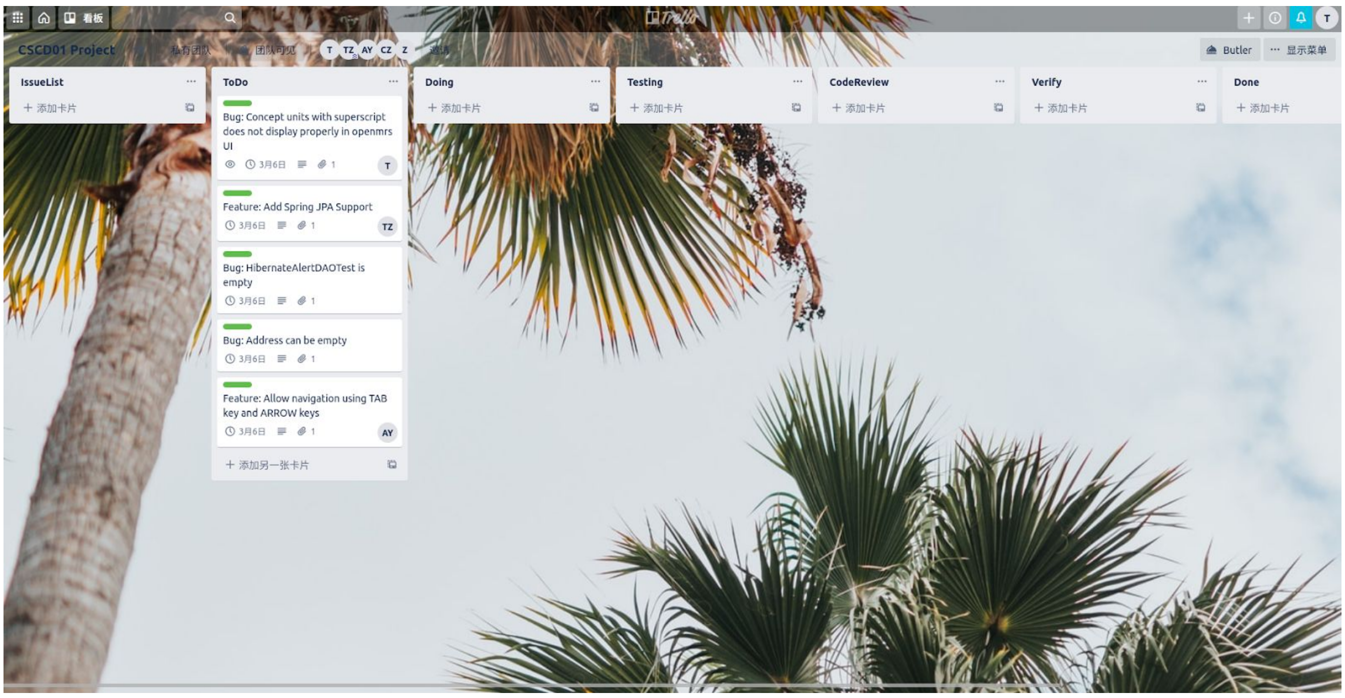
### *0.1.1 Kanban*

Team 21 is using Kanban as our development process. In this deliverable we are going to analyze 5 bugs/features, and choose two of them to fix/add. The team has decided to do them in two different phases: bug analysis phase and bug fix phase.

### Bug analysis phase

In this phase the team is going to give a very detailed explanation of the 5 bugs/features from the issue list.

The following are the screenshots of Kanban board at the bugs analysis phase.



ToDo:

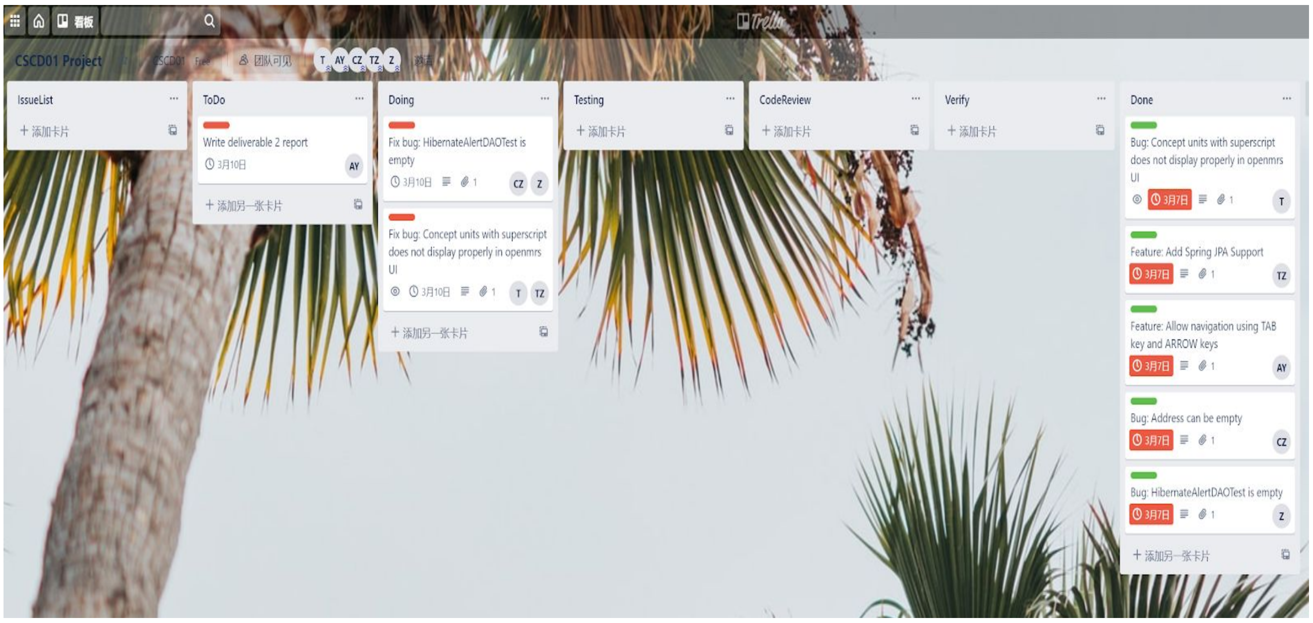
- Bug: Concept units with superscript does not display properly in openmrs UI — 3月6日 — 1 — T
- Feature: Add Spring JPA Support — 3月6日 — 1 — TZ
- Bug: HibernateAlertDAOTest is empty — 3月6日 — 1
- Bug: Address can be empty — 3月6日 — 1
- Feature: Allow navigation using TAB key and ARROW keys — 3月6日 — 1 — AY

*Figure 1.1: Kanban board (at the start of bug analysis phase)*

Each member of the team will be assigned to exactly one task to analyze one chosen bug/feature at the start.

## Bug fix phase

When the team finished the bug analysis phase, we talked about each issue with its estimated fix time and anticipated risks. Finally we made a decision and created another two tasks about the two bugs we will fix during this sprint. In this bug fix phase, two people are going to take care of one bug, including writing test cases and writing a brief technical commentary on the changes that affect the project design, and the one other member is going to finish the report.


CSCD01 Project

IssueList
+ 添加卡片

ToDo
Write deliverable 2 report
3月10日 AY
+ 添加另一张卡片

Doing
Fix bug: HibernateAlertDAOTest is empty
3月10日 1 CZ Z
Fix bug: Concept units with superscript does not display properly in openmrs UI
3月10日 1 T TZ
+ 添加另一张卡片

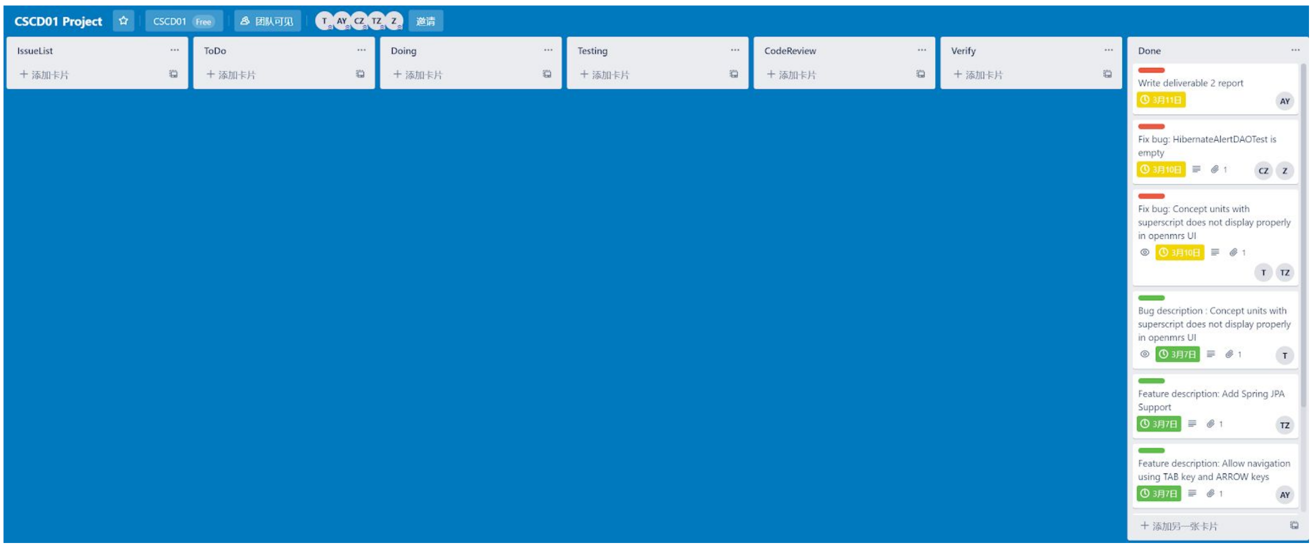
Testing
+ 添加卡片

CodeReview
+ 添加卡片

Verify
+ 添加卡片

Done
Bug: Concept units with superscript does not display properly in openmrs UI
3月7日 1 T
Feature: Add Spring JPA Support
3月7日 1 TZ
Feature: Allow navigation using TAB key and ARROW keys
3月7日 1 AY
Bug: Address can be empty
3月7日 1 CZ
Bug: HibernateAlertDAOTest is empty
3月7日 1 Z
+ 添加另一张卡片

*Figure 1.2: Kanban board (at the start of bug fix phase)*


CSCD01 Project

IssueList
+ 添加卡片

ToDo
+ 添加卡片

Doing
+ 添加卡片

Testing
+ 添加卡片

CodeReview
+ 添加卡片

Verify
+ 添加卡片

Done
Write deliverable 2 report
3月11日 AY
Fix bug: HibernateAlertDAOTest is empty
3月10日 1 CZ Z
Fix bug: Concept units with superscript does not display properly in openmrs UI
3月10日 1 T TZ
Bug description : Concept units with superscript does not display properly in openmrs UI
3月7日 1 T
Feature description: Add Spring JPA Support
3月7日 1 TZ
Feature description: Allow navigation using TAB key and ARROW keys
3月7日 1 AY
+ 添加另一张卡片

*Figure 1.3: Kanban board (at the end of bug fix phase)*

### *0.1.2 Burn up chart*

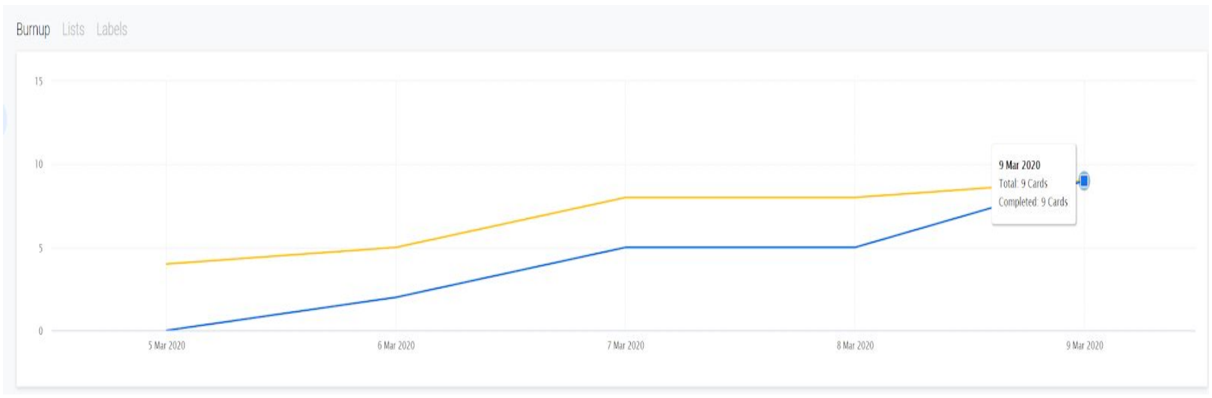


*Figure 1.4: Burn up chart (generate by Corrello)*

### *0.1.3 Standups*

As part of the Kanban process, we are required to do daily standups. However, since we are students and our schedules are not very flexible, we settled on conducting 3 meetings each week.

Mar. 4th: We got everyone up to speed and came up with a plan for the deliverable. The plan is to have each member find and explain one bug/feature. And then have 2 teams of 2 work on 2 bugs/features while the left over one writes the report.

Mar. 6th: We had a meeting to catch up on how everyone was doing with finding and explaining a bug/feature, as the task was due tomorrow noon.

Mar. 7th: We talked about which bugs/features to implement and assigned 2 groups of 2 to work on the two bugs/features. The remaining members will write the report.

## 0.2 Bugs Analysis

In this section, Team 21 will give a clear and detailed explanation of 3 bugs of OpenMRS from the issue list.

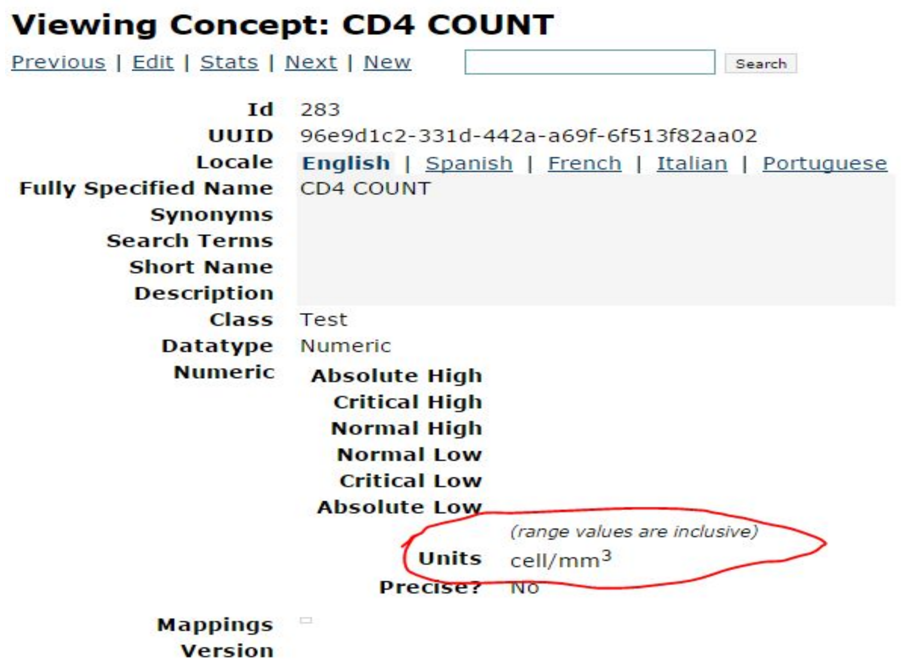
### *0.2.1 Bug Analysis #1*

TRUNK-4762

Concept units with superscript does not display properly in openmrs UI

#### Description

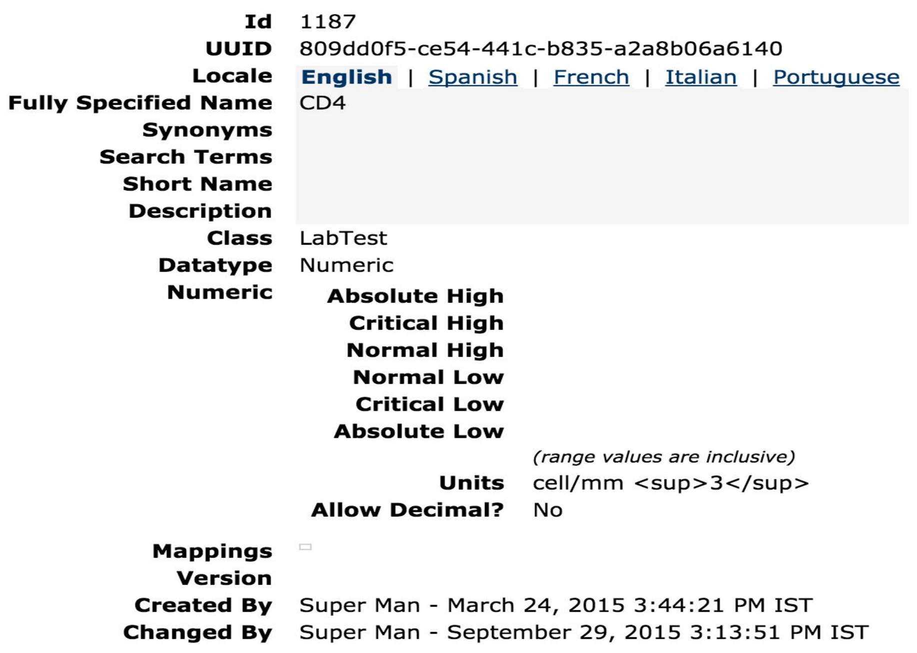
When the concept unit was set as 'cell/mm <sup>3</sup>', it was displayed correctly in 1.10.x. But since 1.11, it is just <sup> tags are displayed as a plain text in UI. Have attached images of displayed units 1.10.x and 1.11.x.


**Viewing Concept: CD4 COUNT**

Previous | Edit | Stats | Next | New    Search

| | |
|---|---|
| **Id** | 283 |
| **UUID** | 96e9d1c2-331d-442a-a69f-6f513f82aa02 |
| **Locale** | **English** \| Spanish \| French \| Italian \| Portuguese |
| **Fully Specified Name** | CD4 COUNT |
| **Synonyms** | |
| **Search Terms** | |
| **Short Name** | |
| **Description** | |
| **Class** | Test |
| **Datatype** | Numeric |
| **Numeric** | **Absolute High** <br> **Critical High** <br> **Normal High** <br> **Normal Low** <br> **Critical Low** <br> **Absolute Low** <br> (range values are inclusive) <br> **Units** cell/mm$^3$ <br> **Precise?** No |
| **Mappings** | |
| **Version** | |

*Figure 2.1: Version 1.10.x (works fine)*



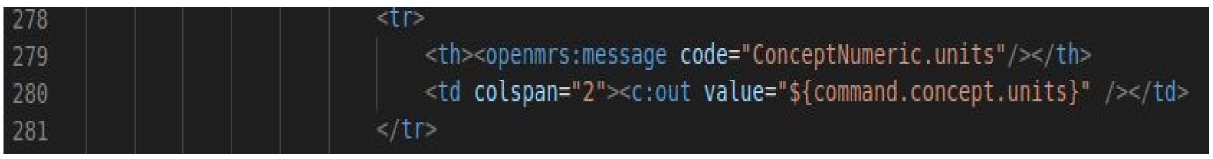
| | | |
|---|---|---|
| **Id** | 1187 | |
| **UUID** | 809dd0f5-ce54-441c-b835-a2a8b06a6140 | |
| **Locale** | **English** \| Spanish \| French \| Italian \| Portuguese | |
| **Fully Specified Name** | CD4 | |
| **Synonyms** | | |
| **Search Terms** | | |
| **Short Name** | | |
| **Description** | | |
| **Class** | LabTest | |
| **Datatype** | Numeric | |
| **Numeric** | **Absolute High** | |
| | **Critical High** | |
| | **Normal High** | |
| | **Normal Low** | |
| | **Critical Low** | |
| | **Absolute Low** | |
| | | *(range values are inclusive)* |
| | **Units** | cell/mm <sup>3</sup> |
| | **Allow Decimal?** | No |
| **Mappings** | | |
| **Version** | | |
| **Created By** | Super Man - March 24, 2015 3:44:21 PM IST | |
| **Changed By** | Super Man - September 29, 2015 3:13:51 PM IST | |

*Figure 2.2: Version 1.11.4 (bug appear)*

## Requirement

The requirement is to fix the bug which when set units as “cell/mm <sup>3</sup>” in the Units bar, the “<sup>3</sup>” will not display as plain text but instead as “cell/mm3”.

## Code Snippet



```
278         <tr>
279             <th><openmrs:message code="ConceptNumeric.units"/></th>
280             <td colspan="2"><c:out value="${command.concept.units}" /></td>
281         </tr>
```

*Figure 2.3: Location of bug*

Referenced openmrs-core module source code files:

Line 278-281, in

openmrs-core/webapp/src/main/webapp/WEB-INF/view/dictionary/concept.jsp

## Analysis Summary

This bug has happened because in the OpenMRS later version 1.11.x there are code updates in the above JSP file which add the <c:out> tag to the Units bar. And <sup> tag inside <c:out> tag will display as plain text instead of a functional tag thus <sup>3</sup> is displayed as plain text in the Units bar.

## Estimated Effort

Basic on the above summary of the bug, we have come up with two possible way of solutions:

1. Delete <c:out> tag at line 280.
2. Use fn:escapeXml() EL function instead of <c:out> tag.

The first solution is easy to apply but the JSP will now be highly risky under XSS attacks, and it will therefore require finding another way to implement XSS prevention. The second solution using fn:escapeXml() EL function to prevent XSS attacks the same as <c:out> tag, but need to rewrite the <tr> raw to support user input. Anyway, the estimated time of our team to fix the bug and do the testing will be 3-5 hours.

## *0.2.2 Bug Analysis #2*

TRUNK-5101

HibernateAlertDAOTest is empty

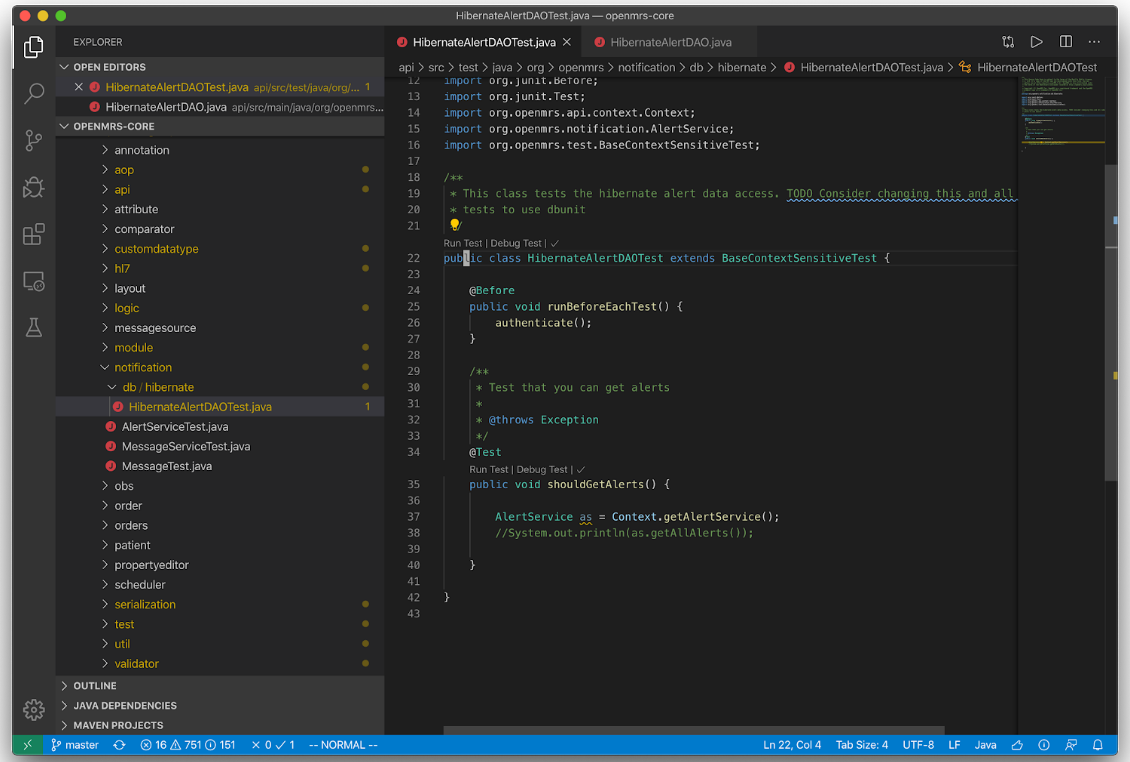
### Description

HibernateAlertDAOTest contains one test with no assertion and some TODO. Add tests or delete it.

### Requirement

The requirement is to add the missing tests to HibernateAlertADOTest.

### Code Snippet



```
12  import org.junit.Before;
13  import org.junit.Test;
14  import org.openmrs.api.context.Context;
15  import org.openmrs.notification.AlertService;
16  import org.openmrs.test.BaseContextSensitiveTest;
17
18  /**
19   * This class tests the hibernate alert data access. TODO Consider changing this and all
20   * tests to use dbunit
21   */
22  public class HibernateAlertDAOTest extends BaseContextSensitiveTest {
23
24      @Before
25      public void runBeforeEachTest() {
26          authenticate();
27      }
28
29      /**
30       * Test that you can get alerts
31       *
32       * @throws Exception
33       */
34      @Test
35      public void shouldGetAlerts() {
36
37          AlertService as = Context.getAlertService();
38          //System.out.println(as.getAllAlerts());
39
40      }
41
42  }
43
```

*Figure 2.4: Location of bug*

Line 22,

api/src/test/java/org/openmrs/notification/db/hibernate/HibernateAlertDAOTest.java

## Analysis Summary

This bug is due to the use of #TODO tag, was introduced in 2017. The reason for the bug is due to missing tests. A great portion of the code remains untested in this project, therefore the #TODO tag is potentially abused. This is potentially dangerous, since we cannot ensure the quality of untested code.

## Estimated Effort

Basic on the above summary of the bug, we have come up with two possible way of solutions:

1. Delete the test skeleton completely
2. Add missing test cases to the code

The first approach would be easier. However, since most code under api/src/main/java/org/openmrs/notification/db/hibernate is untested, we would ensure better software quality by adding the tests. It is always good to aim for better test coverage. The estimated time taken to fix this bug would be ~10 hours.

## *0.2.3 Bug Analysis #3*

RA-495

Address can be empty

### Description

The address could be deleted by editing contact info. The address should not be empty when editing the field. This can be produced by show contact info → edit (address) → empty the address → Confirm, yes.

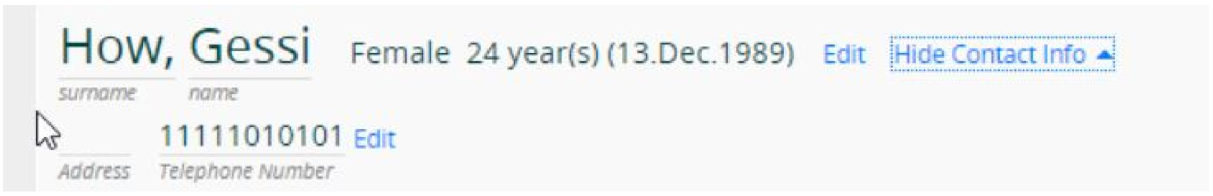
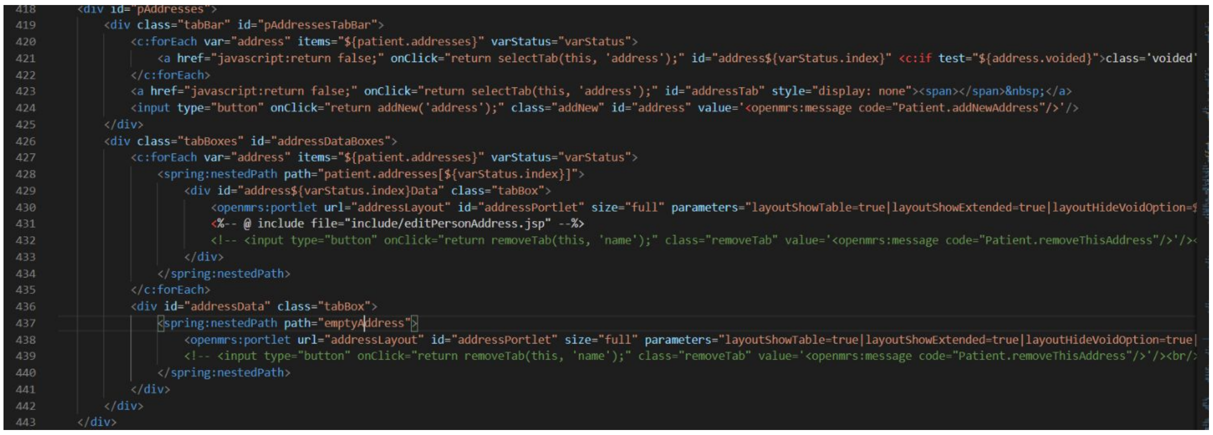


*Figure 2.5: How the bug looks like*

### Requirements

There are two approaches to solve this bug, one is from frontend and the other one is the backend. From the frontend we have to make sure the user entered legitimate input. From the backend we have to make sure the user's data is valid.

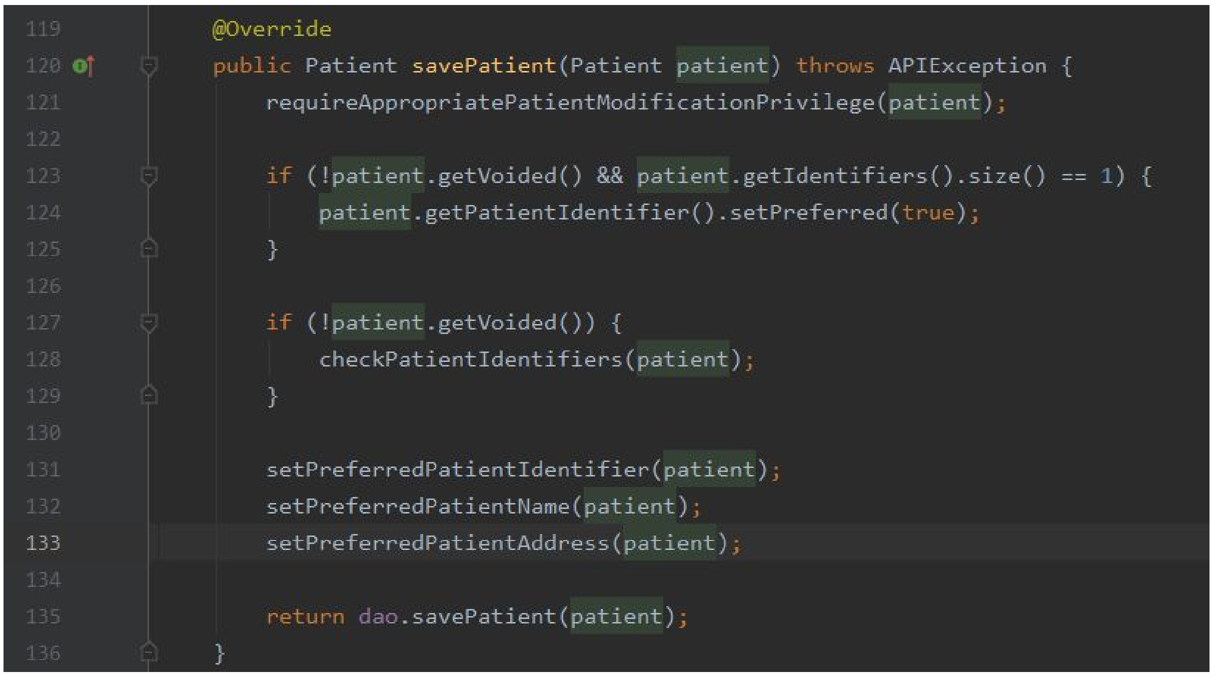
### Code Snippet



```
<div id="pAddresses">
    <div class="tabBar" id="pAddressesTabBar">
        <c:forEach var="address" items="${patient.addresses}" varStatus="varStatus">
            <a href="javascript:return false;" onClick="return selectTab(this, 'address');" id="address${varStatus.index}" <c:if test="${address.voided}">class='voided'
        </c:forEach>
        <a href="javascript:return false;" onClick="return selectTab(this, 'address');" id="addressTab" style="display: none"><span></span>&nbsp;</a>
        <input type="button" onClick="return addNew('address');" class="addNew" id="address" value='<openmrs:message code="Patient.addNewAddress"/>'/>
    </div>
    <div class="tabBoxes" id="addressDataBoxes">
        <c:forEach var="address" items="${patient.addresses}" varStatus="varStatus">
            <spring:nestedPath path="patient.addresses[${varStatus.index}]">
                <div id="address${varStatus.index}Data" class="tabBox">
                    <openmrs:portlet url="addressLayout" id="addressPortlet" size="full" parameters="layoutShowTable=true|layoutShowExtended=true|layoutHideVoidOption=$
                    <%-- @ include file="include/editPersonAddress.jsp" --%>
                    <!-- <input type="button" onClick="return removeTab(this, 'name');" class="removeTab" value='<openmrs:message code="Patient.removeThisAddress"/>'/><
                </div>
            </spring:nestedPath>
        </c:forEach>
        <div id="addressData" class="tabBox">
            <spring:nestedPath path="emptyAddress">
                <openmrs:portlet url="addressLayout" id="addressPortlet" size="full" parameters="layoutShowTable=true|layoutShowExtended=true|layoutHideVoidOption=true|
                <!-- <input type="button" onClick="return removeTab(this, 'name');" class="removeTab" value='<openmrs:message code="Patient.removeThisAddress"/>'/><br/>
            </spring:nestedPath>
        </div>
    </div>
</div>
```

*Figure 2.6: Frontend code cause the bug*

From the file patientFrom.jsp, line 418 to line 443, the divs handle the address input part.



```
119         @Override
120         public Patient savePatient(Patient patient) throws APIException {
121             requireAppropriatePatientModificationPrivilege(patient);
122 
123             if (!patient.getVoided() && patient.getIdentifiers().size() == 1) {
124                 patient.getPatientIdentifier().setPreferred(true);
125             }
126 
127             if (!patient.getVoided()) {
128                 checkPatientIdentifiers(patient);
129             }
130 
131             setPreferredPatientIdentifier(patient);
132             setPreferredPatientName(patient);
133             setPreferredPatientAddress(patient);
134 
135             return dao.savePatient(patient);
136         }
```

*Figure 2.7: Backend code causes the bug*

The backend code causes the bug is from api/impl/PatientServiceImpl.java. Line 133, adding the address.

## Analysis Summary

The frontend part does have a lot of divs to handle the input of addresses. Although there is a div that handles the empty address, but it does nothing with it, just pass the case. So this causes the empty address bug.
The backend part is an implementation of the PatientService. This file connects with the database access object then modify the database through it. In this method savePatient, it doesn't check if the address is legal or not, it just passes the data to the database, and this will cause the empty address bug.

## Estimated Effort

Based on these two approaches and the analysis. It's very clear to fix the bug, from the frontend, we can add some restrictions to make sure the user entered the legal input. From the backend, we can add some confirmation functions that make sure the user entered legal input before update or insert the data into the database.

## Time Estimation

For the frontend, the JSP is a completely new world for us, it might take 2 or 3 days to learn the JSP from the beginning and need 1 more day to complete the fix. So the total might be 4 days for a frontend fix. On the other hand, the backend part is very comfortable for us, it takes 1 day to figure out how the service interacts with the database and need 1 more day to fix the bug. So it might take 2 days for us to fix the bug from the backend.

## 0.3 Feature Analysis

In this section, we will discuss what features we are going to add.

### *0.3.1 Feature Analysis #1*

TRUNK-4667: Add Spring JPA Support

#### Description

Add Spring JPA to improve searching data in the database. It can connect with Hibernate to perform a searching method of Object-oriented rather than database-oriented.

#### Requirement

Import the JPA library so that other classes will search for data such as a patient or a specific order easily.

#### Analysis Summary

This feature should be added to the Encounter class from the database layer. As we discussed in deliverable 1, the Encounter class has high interaction with the Patient class (figure 3.1). It will do a lot of jobs searching for a patient to interact with other classes. For instance, find a drug order for a specific patient. Therefore, it is important for developers to implement JPA to search for data between Java objects/classes and a relational database. Overall, we need to implement a new DAO in the EncounterDAO.java file under the database package (figure 3.2).

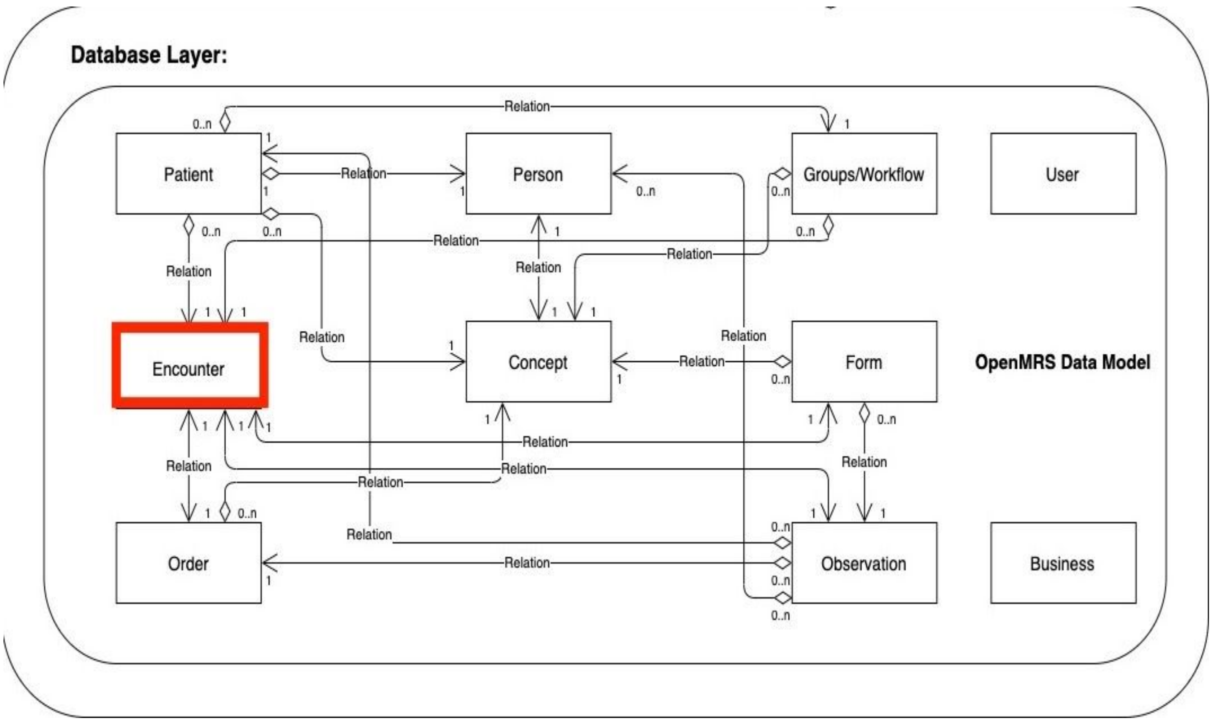


Figure 3.1: Database Layer

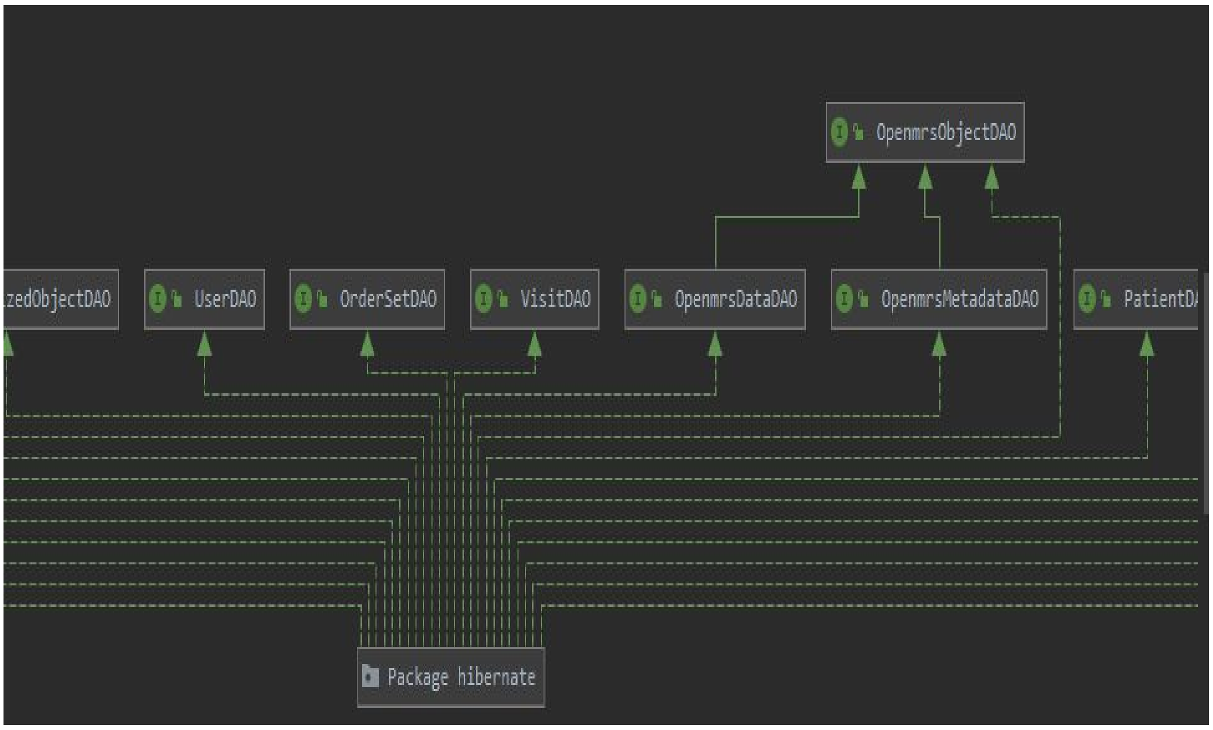
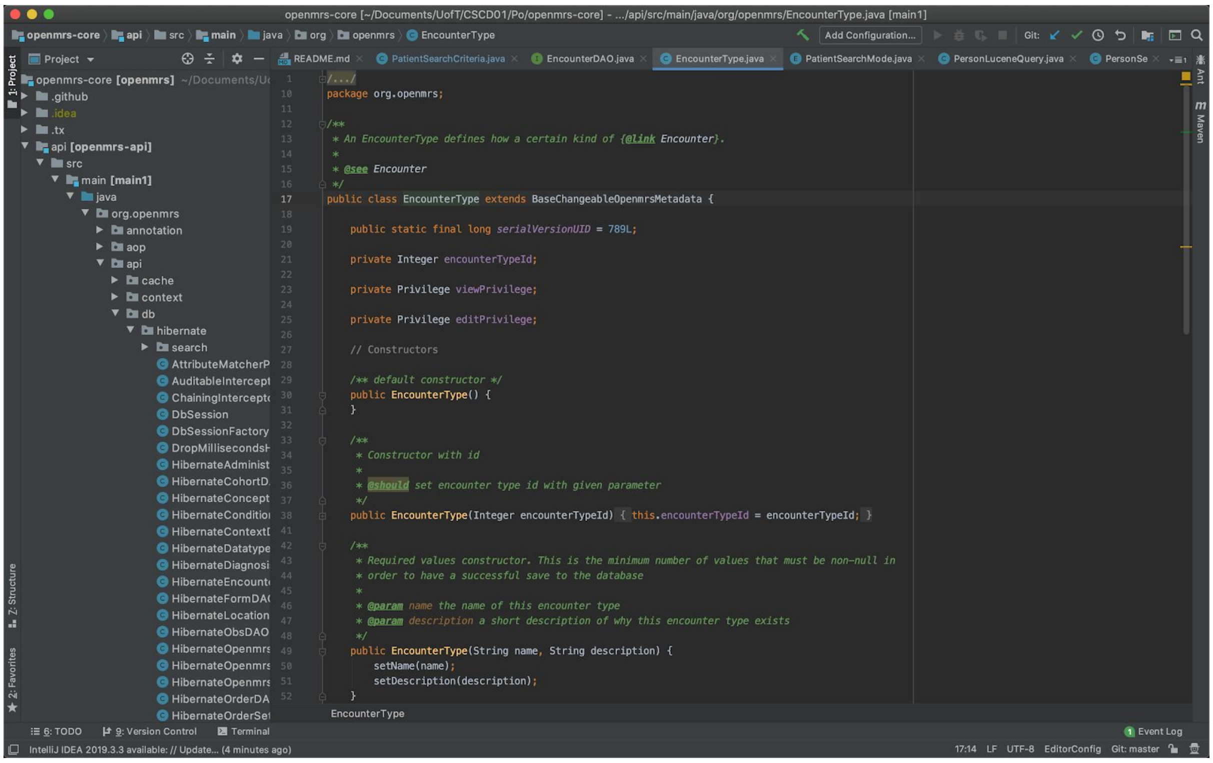


Figure 3.2: UML for the database package

## Code Snippet

Under /openmrs-core/api/src/main/java/org/openmrs/api/db/EncounterDAO.java



```java
/.../
package org.openmrs;

/**
 * An EncounterType defines how a certain kind of {@link Encounter}.
 *
 * @see Encounter
 */
public class EncounterType extends BaseChangeableOpenmrsMetadata {

	public static final long serialVersionUID = 789L;

	private Integer encounterTypeId;

	private Privilege viewPrivilege;

	private Privilege editPrivilege;

	// Constructors

	/** default constructor */
	public EncounterType() {
	}

	/**
	 * Constructor with id
	 *
	 * @should set encounter type id with given parameter
	 */
	public EncounterType(Integer encounterTypeId) { this.encounterTypeId = encounterTypeId; }

	/**
	 * Required values constructor. This is the minimum number of values that must be non-null in
	 * order to have a successful save to the database
	 *
	 * @param name the name of this encounter type
	 * @param description a short description of why this encounter type exists
	 */
	public EncounterType(String name, String description) {
		setName(name);
		setDescription(description);
	}
```

*Figure 3.3: Code snippet*

## Time Estimation

The difficulty of adding this feature is there might be code in other classes who are related to the Encounter class that needs to change also in the database. So, fixing other problems to suit this new feature will take a lot of time. Thus, our estimated time to add this feature is 4 days.

## *0.3.2 Feature Analysis #2*

XFRM-205: Allow navigation using TAB key and ARROW keys

### Description

When navigating a form with select input using the tab key, it sticks on the input, unless the ENTER key or one of the ARROW keys are pressed. It would be preferable if one would navigate away from the select input using the TAB key too.

On the other hand, using arrow keys to navigate away from select input works, but it also changes the current selection.

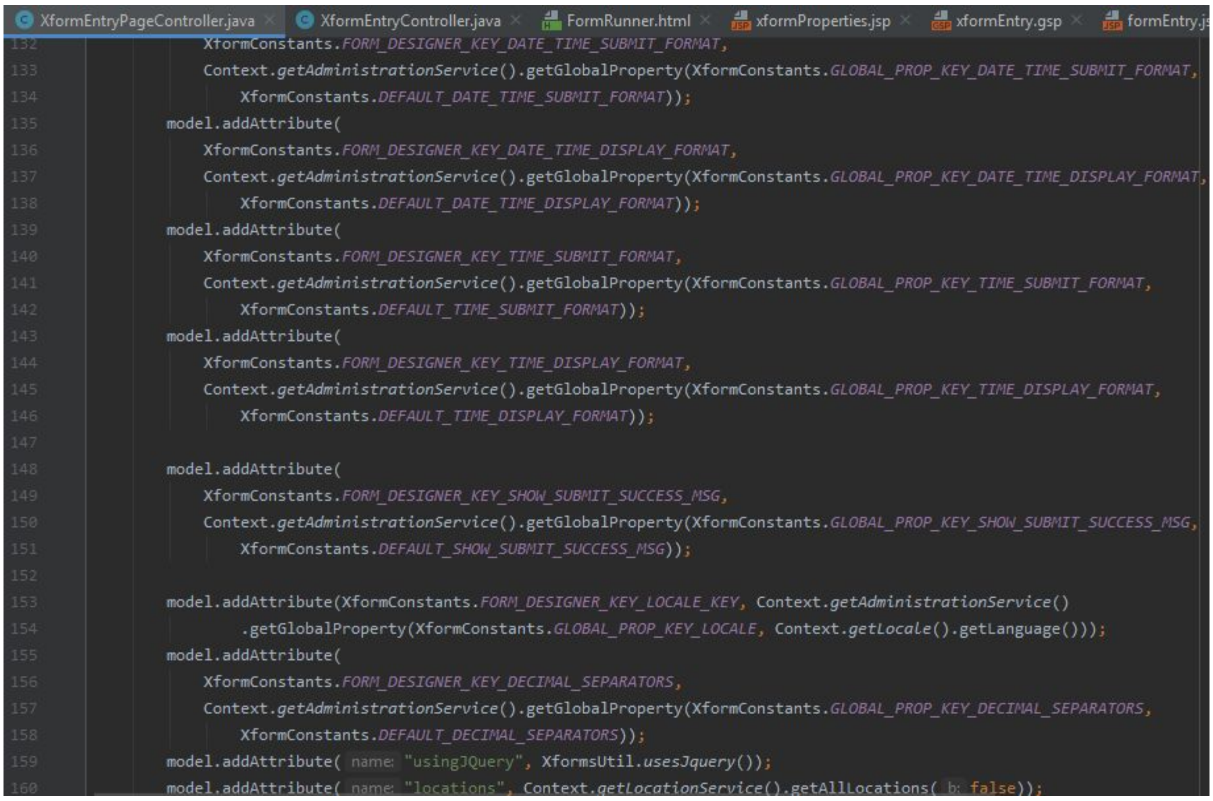
### Requirement

Add functionality in the code so that when the tab key is pressed, the select input is changed.

### Analysis Summary

Relevant files to XForm entry are XformEntryPageController, XformEntryController.java, xformEntry.gsp, formEntry.jsp, FormDesigner.html, xformProperties.jsp. Look through these files to see how each of these files contributes to designing the form and where would be appropriate to implement the new feature. Research how to adjust the tab functionality and apply it.

## Code Snippet



```java
            XformConstants.FORM_DESIGNER_KEY_DATE_TIME_SUBMIT_FORMAT,
            Context.getAdministrationService().getGlobalProperty(XformConstants.GLOBAL_PROP_KEY_DATE_TIME_SUBMIT_FORMAT,
                XformConstants.DEFAULT_DATE_TIME_SUBMIT_FORMAT));
        model.addAttribute(
            XformConstants.FORM_DESIGNER_KEY_DATE_TIME_DISPLAY_FORMAT,
            Context.getAdministrationService().getGlobalProperty(XformConstants.GLOBAL_PROP_KEY_DATE_TIME_DISPLAY_FORMAT,
                XformConstants.DEFAULT_DATE_TIME_DISPLAY_FORMAT));
        model.addAttribute(
            XformConstants.FORM_DESIGNER_KEY_TIME_SUBMIT_FORMAT,
            Context.getAdministrationService().getGlobalProperty(XformConstants.GLOBAL_PROP_KEY_TIME_SUBMIT_FORMAT,
                XformConstants.DEFAULT_TIME_SUBMIT_FORMAT));
        model.addAttribute(
            XformConstants.FORM_DESIGNER_KEY_TIME_DISPLAY_FORMAT,
            Context.getAdministrationService().getGlobalProperty(XformConstants.GLOBAL_PROP_KEY_TIME_DISPLAY_FORMAT,
                XformConstants.DEFAULT_TIME_DISPLAY_FORMAT));

        model.addAttribute(
            XformConstants.FORM_DESIGNER_KEY_SHOW_SUBMIT_SUCCESS_MSG,
            Context.getAdministrationService().getGlobalProperty(XformConstants.GLOBAL_PROP_KEY_SHOW_SUBMIT_SUCCESS_MSG,
                XformConstants.DEFAULT_SHOW_SUBMIT_SUCCESS_MSG));

        model.addAttribute(XformConstants.FORM_DESIGNER_KEY_LOCALE_KEY, Context.getAdministrationService()
                .getGlobalProperty(XformConstants.GLOBAL_PROP_KEY_LOCALE, Context.getLocale().getLanguage()));
        model.addAttribute(
            XformConstants.FORM_DESIGNER_KEY_DECIMAL_SEPARATORS,
            Context.getAdministrationService().getGlobalProperty(XformConstants.GLOBAL_PROP_KEY_DECIMAL_SEPARATORS,
                XformConstants.DEFAULT_DECIMAL_SEPARATORS));
        model.addAttribute( name: "usingJQuery", XformsUtil.usesJquery());
        model.addAttribute( name: "locations", Context.getLocationService().getAllLocations( b: false));
```

*Figure 3.4: Code snippet*

## Time Estimation

We need time to look through the architecture of the module, including the related files and figure out what we need to do to make the tab key work. Taking researching and continuous testing into account, we estimate it will take about 2-3 days to complete this feature.

## 0.4 Items to Implement

We have decided to implement the bugs detailed in 0.2.1 and 0.2.2.

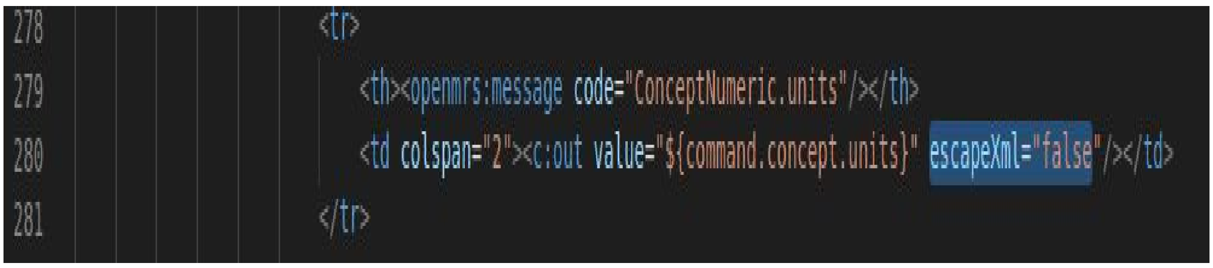
### *0.4.1 TRUNK-4762 Fix and Test*

#### Reason we chose TRUNK-4762

After we finish the bug analysis phase, we have only 2 days til the deadline. And so we were reminded the reason why we chose Kanban as our development process, which is that we care more about quality than quantity. With only 2 days for the bug fix phase, we want to deliver the best quality work, to make sure the bug is fixed. Therefore, the TRUNK-4762 is one of the best choices. Besides, the TRUNK-4762 is a display bug, which is easy to fix but hard to locate where the bug happens. Luckily, we already did more than half work when we discovered the location of the bug. There is also little to no risks to implementing this fix since it is just a display bug and we can easily revert. Thus, choosing TRUNK-4762 to fix is reasonable for us.

#### How to fix

We add “escapeXml=false” into line 280 of concept.jsp to make sure the Units bar does not escape special XML tag characters.



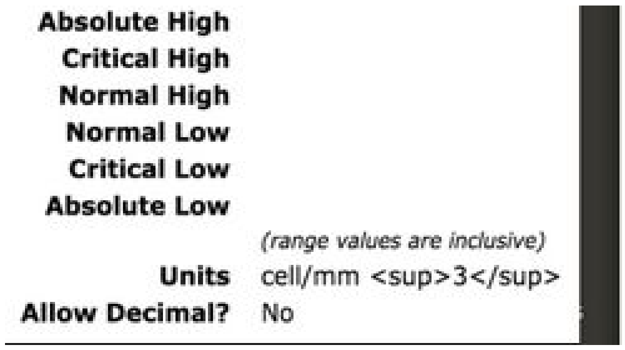
```
278                 <tr>
279                     <th><openmrs:message code="ConceptNumeric.units"/></th>
280                     <td colspan="2"><c:out value="${command.concept.units}" escapeXml="false"/></td>
281                 </tr>
```

*Figure 4.1: concept.jsp after fix*

Location of target file:

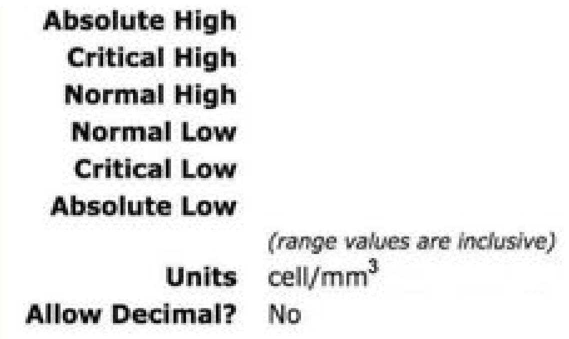
openmrs-core/webapp/src/main/webapp/WEB-INF/view/dictionary/concept.jsp

Before fix:



| | |
|---|---|
| Absolute High | |
| Critical High | |
| Normal High | |
| Normal Low | |
| Critical Low | |
| Absolute Low | |
| | *(range values are inclusive)* |
| Units | cell/mm <sup>3</sup> |
| Allow Decimal? | No |

After fix:



| | |
|---|---|
| Absolute High | |
| Critical High | |
| Normal High | |
| Normal Low | |
| Critical Low | |
| Absolute Low | |
| | *(range values are inclusive)* |
| Units | cell/mm$^3$ |
| Allow Decimal? | No |

*Figure 4.2: Before and after fix*

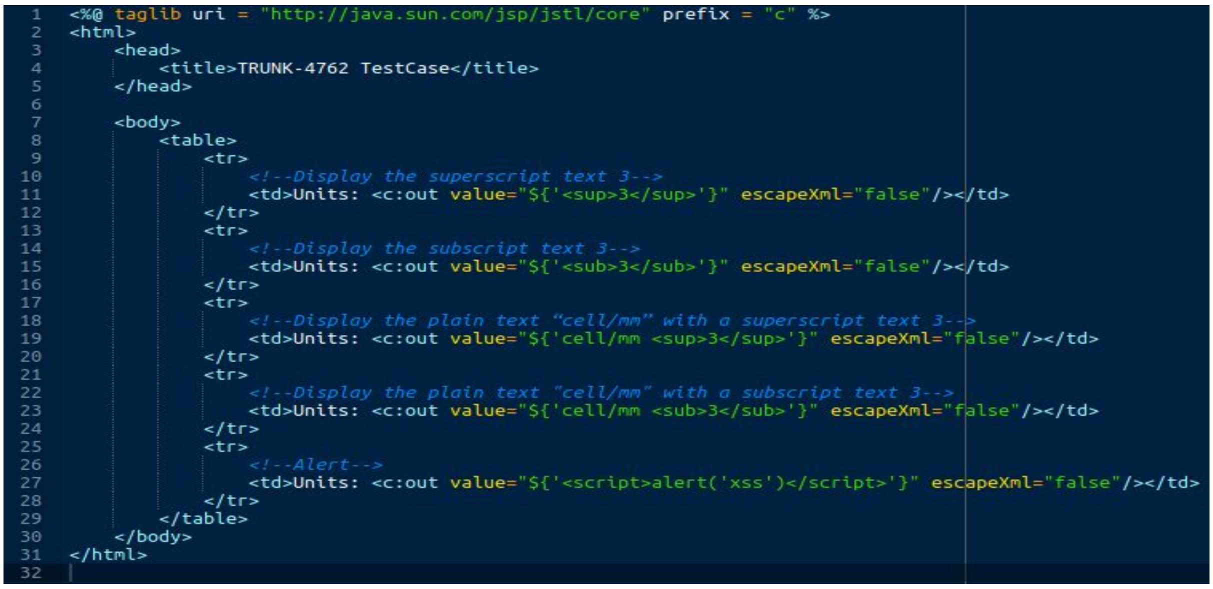
This bug fix is only one line change inside openmrs-core, so it does not change any design or structure of the code.

## Acceptance Tests

Since the bug is a display bug and happened in a jsp file with no user input, there is no way we can write unit tests for a jsp file. Thus, we decided to write acceptance tests. We have design a series of examples as following:

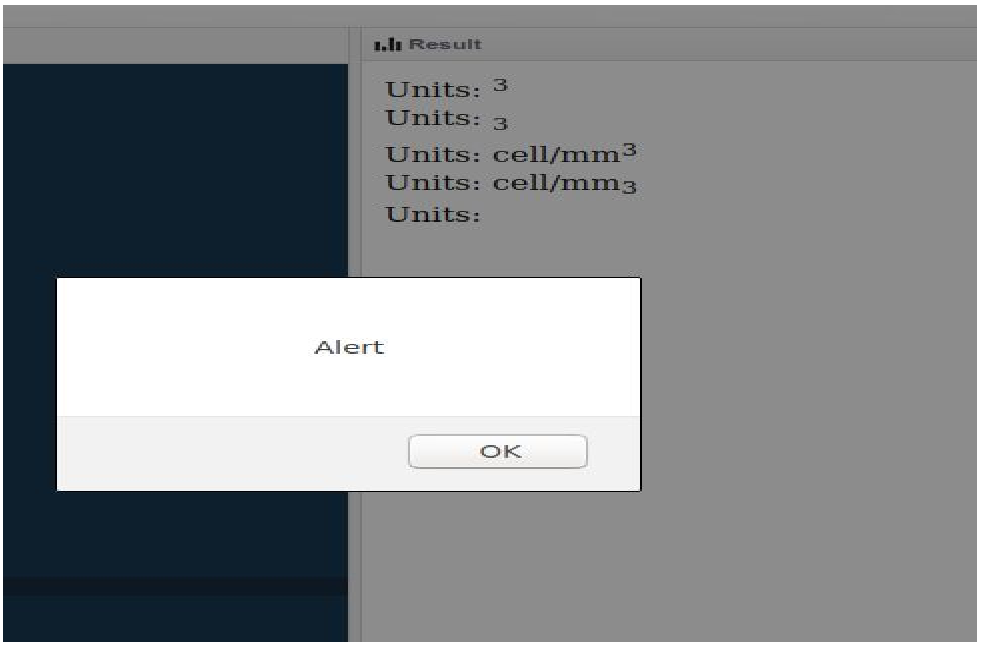
1. If “<sup>3</sup>” is the text inside the Units bar, then the bar should display as superscript text 3, not plain text.
2. If “<sub>3</sub>” is the text inside the Units bar, then the bar should display as subscript text 3, not plain text.
3. If “cell/mm <sup>3</sup>” is inside the Units bar, the bar should display as plain text “cell/mm” with a superscript text 3, together should be “cell/mm$^3$”.
4. If “cell/mm <sub>3</sub>” inside the Units bar, then the bar should display as plain text “cell/mm” with a subscript text 3, together should be “cell/mm$_3$”.
5. If “<script>alert('xss')</script>” inside the Units bar, then the bar should come up with an alert.

The above test has one common purpose, which is to make sure that any special XML tag characters that are inside the Units bar, will display as functional tags, not simple plain text. And since we can not write code to auto fill in the Units bar, we decided to write a text jsp file to “simulate” the Units bar. Here is our test code and its result looks like:



```
<%@ taglib uri = "http://java.sun.com/jsp/jstl/core" prefix = "c" %>
<html>
    <head>
        <title>TRUNK-4762 TestCase</title>
    </head>

    <body>
        <table>
            <tr>
                <!--Display the superscript text 3-->
                <td>Units: <c:out value="${'<sup>3</sup>'}" escapeXml="false"/></td>
            </tr>
            <tr>
                <!--Display the subscript text 3-->
                <td>Units: <c:out value="${'<sub>3</sub>'}" escapeXml="false"/></td>
            </tr>
            <tr>
                <!--Display the plain text “cell/mm” with a superscript text 3-->
                <td>Units: <c:out value="${'cell/mm <sup>3</sup>'}" escapeXml="false"/></td>
            </tr>
            <tr>
                <!--Display the plain text "cell/mm" with a subscript text 3-->
                <td>Units: <c:out value="${'cell/mm <sub>3</sub>'}" escapeXml="false"/></td>
            </tr>
            <tr>
                <!--Alert-->
                <td>Units: <c:out value="${'<script>alert('xss')</script>'}" escapeXml="false"/></td>
            </tr>
        </table>
    </body>
</html>
```

*Figure 4.3: Codes of test file*



Units: $^{3}$
Units: $_{3}$
Units: cell/mm$^{3}$
Units: cell/mm$_{3}$
Units:

*Figure 4.4: Result of test file*

The result here shows that, after fix, any special XML tag characters that inside c:out tag are now functional.

## Instruction on how to run the test

If you would like to do the test within actual OpenMRS program, the only way to do is download the modified concept.jsp file, and swap with the file at: openmrs-core/webapp/src/main/webapp/WEB-INF/view/dictionary/concept.jsp. Also, make sure your openmrs-core version is 1.11.4. After that, execute:

1. cd openmrs-core
2. mvn clean package
3. cd openmrs-core/webapp
4. mvn jetty:run

Then you can access OpenMRS at localhost:8080/openmrs. Finally you can go to the page and see the information on Units bar is now displayed normally.

If you wish to run the test on the simulate Units bar, download testCase.jsp, and install web service like Apache Tomcat, then run it. All the test cases contain exactly the same code as in concept.jsp, thus the output of test will be the same display on OpenMRS software after fix.

## *0.4.2 TRUNK-5101 Fix and Test*

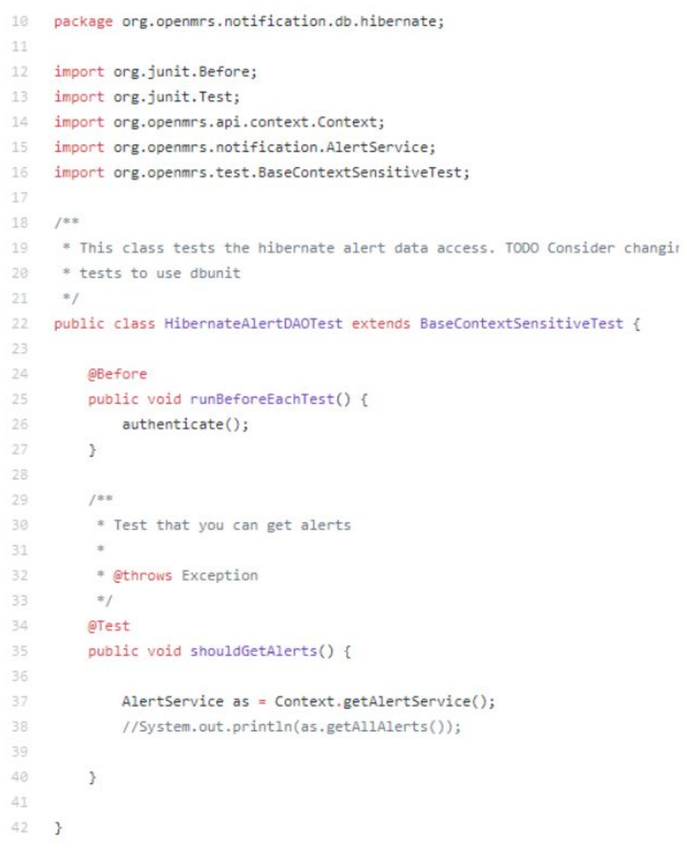
### Reason we chose TRUNK-5101

We chose this bug to fix because we think it can be easily done within the time constraints we have since all we are doing is writing some test cases. Besides, the priority of this issue is on the highest level; untested code can be dangerous so adding test cases would benefit the overall quality of the program. Writing these test cases would also help us better understand the software’s functionality. Therefore this is a great choice for us to implement.

### How to fix

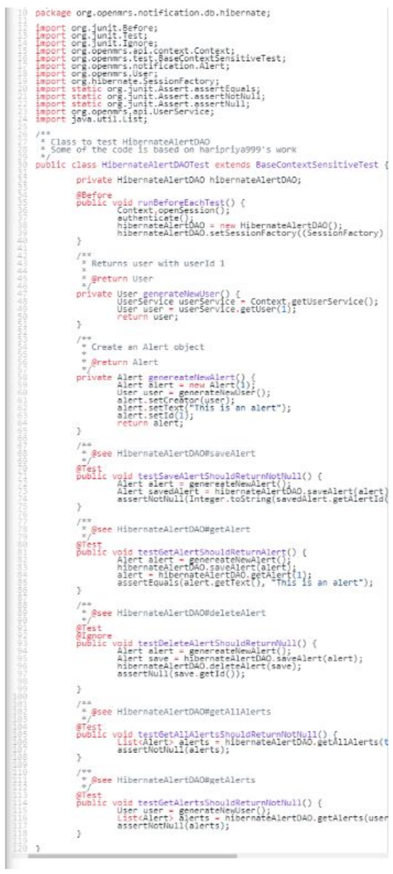
We fixed this issue by writing a complete test suite in HibernateAlertDAOTest.java. We explore all cases and edge cases of HibernateAlertADO and write appropriate unit tests for them.

Before fix:



```java
package org.openmrs.notification.db.hibernate;

import org.junit.Before;
import org.junit.Test;
import org.openmrs.api.context.Context;
import org.openmrs.notification.AlertService;
import org.openmrs.test.BaseContextSensitiveTest;

/**
 * This class tests the hibernate alert data access. TODO Consider changi
 * tests to use dbunit
 */
public class HibernateAlertDAOTest extends BaseContextSensitiveTest {

    @Before
    public void runBeforeEachTest() {
        authenticate();
    }

    /**
     * Test that you can get alerts
     *
     * @throws Exception
     */
    @Test
    public void shouldGetAlerts() {

        AlertService as = Context.getAlertService();
        //System.out.println(as.getAllAlerts());

    }

}
```

After fix:



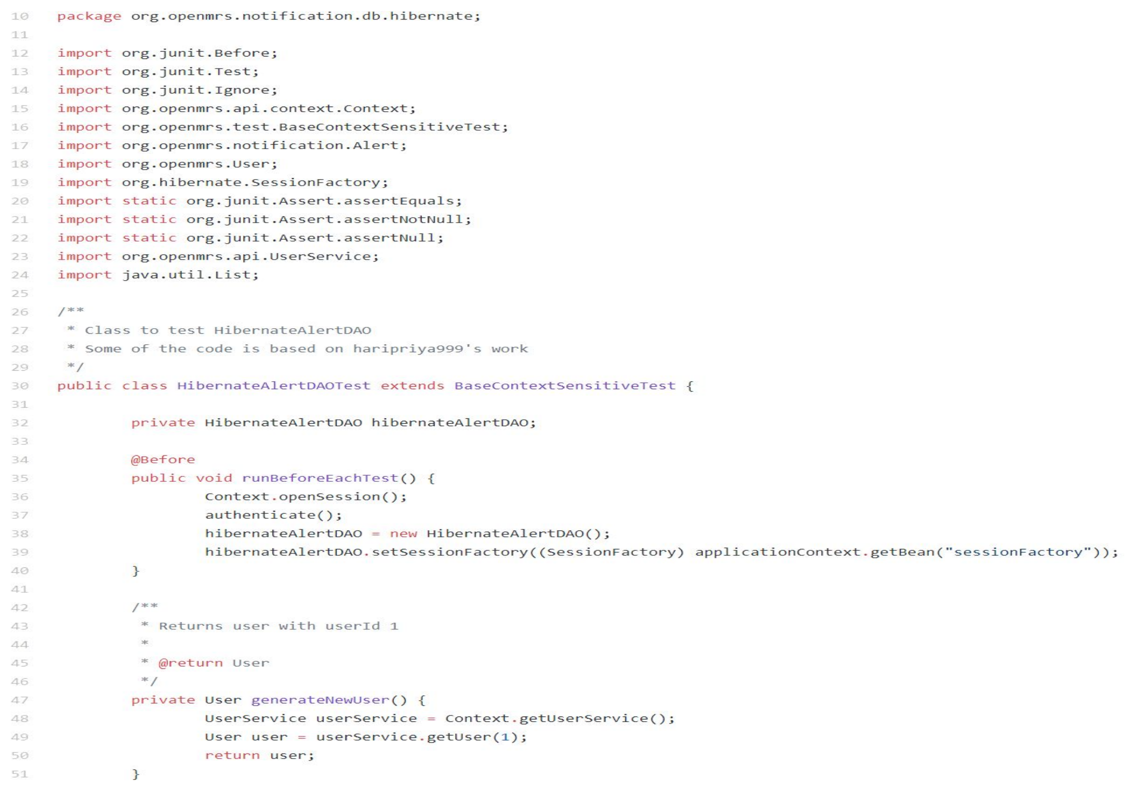
```java
package org.openmrs.notification.db.hibernate;

import org.junit.Before;
import org.junit.Test;
import org.junit.Ignore;
import org.openmrs.api.context.Context;
import org.openmrs.test.BaseContextSensitiveTest;
import org.openmrs.notification.Alert;
import org.openmrs.User;
import org.hibernate.SessionFactory;
import static org.junit.Assert.assertEquals;
import static org.junit.Assert.assertNotNull;
import static org.junit.Assert.assertNull;
import org.openmrs.api.UserService;
import java.util.List;

/**
 * Class to test HibernateAlertDAO
 * Some of the code is based on haripriya999's work
 */
public class HibernateAlertDAOTest extends BaseContextSensitiveTest {

        private HibernateAlertDAO hibernateAlertDAO;

        @Before
        public void runBeforeEachTest() {
                Context.openSession();
                authenticate();
                hibernateAlertDAO = new HibernateAlertDAO();
                hibernateAlertDAO.setSessionFactory((SessionFactory)
        }

        /**
         * Returns user with userId 1
         *
         * @return User
         */
        private User generateNewUser() {
                UserService userService = Context.getUserService();
                User user = userService.getUser(1);
                return user;
        }

        /**
         * Create an Alert object
         *
         * @return Alert
         */
        private Alert genereateNewAlert() {
                Alert alert = new Alert(1);
                User user = generateNewUser();
                alert.setCreator(user);
                alert.setText("This is an alert");
                alert.setId(1);
                return alert;
        }

        /**
         * @see HibernateAlertDAO#saveAlert
         */
        @Test
        public void testSaveAlertShouldReturnNotNull() {
                Alert alert = genereateNewAlert();
                Alert savedAlert = hibernateAlertDAO.saveAlert(alert)
                assertNotNull(Integer.toString(savedAlert.getAlertId(
        }

        /**
         * @see HibernateAlertDAO#getAlert
         */
        @Test
        public void testGetAlertShouldReturnAlert() {
                Alert alert = genereateNewAlert();
                hibernateAlertDAO.saveAlert(alert);
                alert = hibernateAlertDAO.getAlert(1);
                assertEquals(alert.getText(), "This is an alert");
        }

        /**
         * @see HibernateAlertDAO#deleteAlert
         */
        @Test
        @Ignore
        public void testDeleteAlertShouldReturnNull() {
                Alert alert = genereateNewAlert();
                Alert save = hibernateAlertDAO.saveAlert(alert);
                hibernateAlertDAO.deleteAlert(save);
                assertNull(save.getId());

        }

        /**
         * @see HibernateAlertDAO#getAllAlerts
         */
        @Test
        public void testGetAllAlertsShouldReturnNotNull() {
                List<Alert> alerts = hibernateAlertDAO.getAllAlerts(t
                assertNotNull(alerts);
        }

        /**
         * @see HibernateAlertDAO#getAlerts
         */
        @Test
        public void testGetAlertsShouldReturnNotNull() {
                User user = generateNewUser();
                List<Alert> alerts = hibernateAlertDAO.getAlerts(user
                assertNotNull(alerts);
        }

}
```

*Figure 4.5: HibernateAlertDAOTest.java (left before fix, right after fix)*

## Acceptance Tests

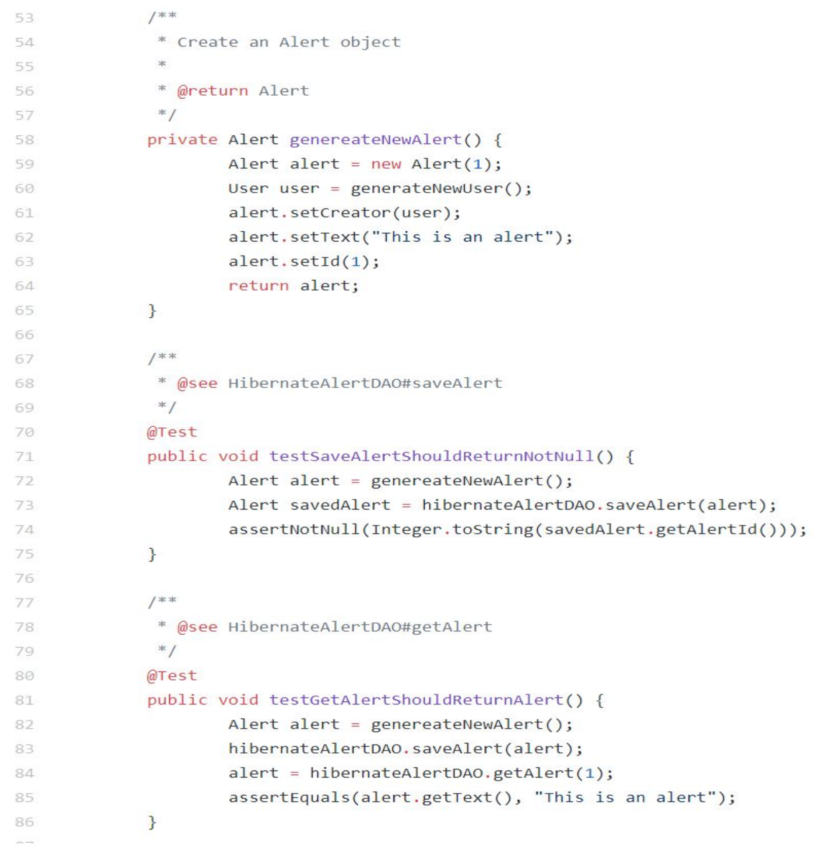
Since all we are doing is writing test cases, we cannot “write test for a test” so there are no test cases to demonstrate the correctness of changes, since the only change is in the test file and there is no actual code behaviour changed. However, for adding test cases for HibernateAlertDAOTest.java, we have designed a series of cases as customer “acceptance tests”, as following:

1. When the user creates a new alert and the alert is saved by the system, then the alert list should not be empty.
2. When the user wants to get an alert from the system, then the return result should be an alert.
3. When the user delete the only alert from the system, then the alert list should be empty.
4. When the user wants to get all alerts from the system (suppose alert list is not empty), then the result should not be null.
5. When the user wants to get all alerts created by a specific user from the system (suppose alert list is not empty), then the result should not be null.



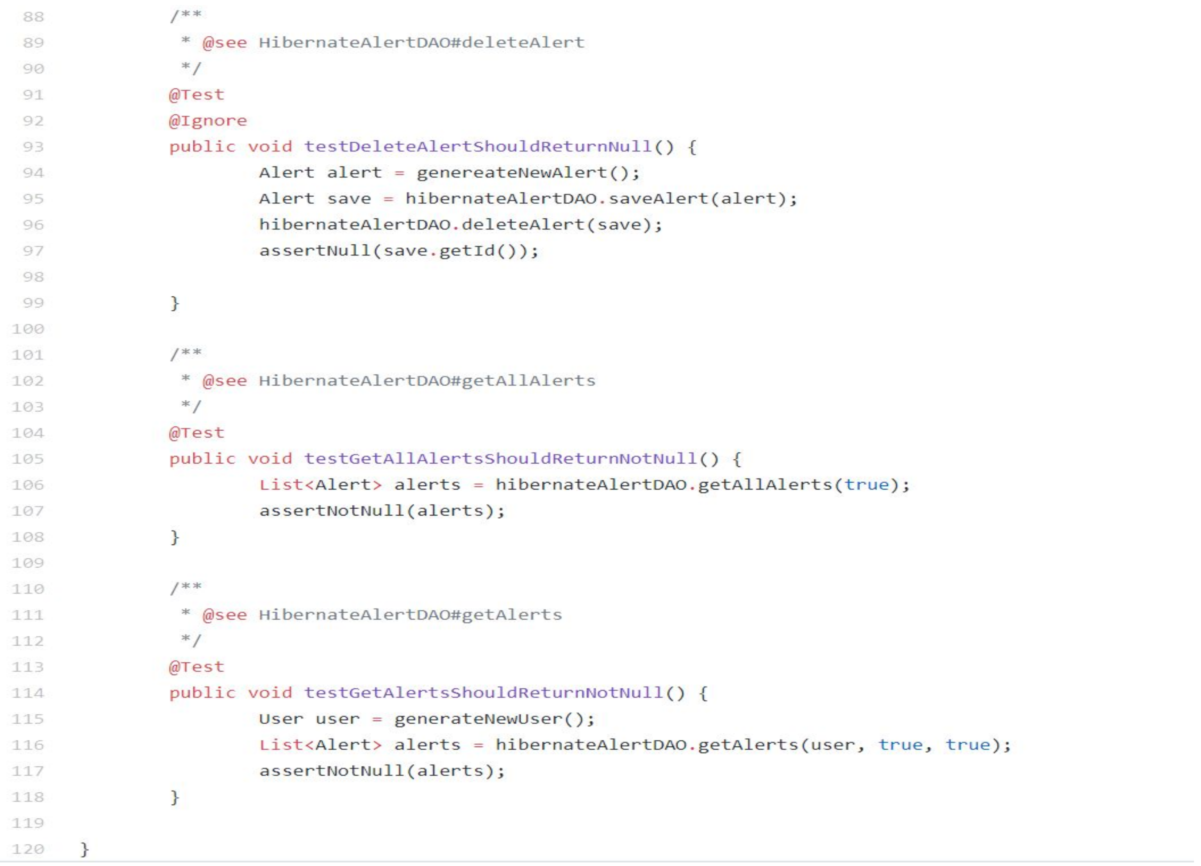
```java
package org.openmrs.notification.db.hibernate;

import org.junit.Before;
import org.junit.Test;
import org.junit.Ignore;
import org.openmrs.api.context.Context;
import org.openmrs.test.BaseContextSensitiveTest;
import org.openmrs.notification.Alert;
import org.openmrs.User;
import org.hibernate.SessionFactory;
import static org.junit.Assert.assertEquals;
import static org.junit.Assert.assertNotNull;
import static org.junit.Assert.assertNull;
import org.openmrs.api.UserService;
import java.util.List;

/**
 * Class to test HibernateAlertDAO
 * Some of the code is based on haripriya999's work
 */
public class HibernateAlertDAOTest extends BaseContextSensitiveTest {

        private HibernateAlertDAO hibernateAlertDAO;

        @Before
        public void runBeforeEachTest() {
                Context.openSession();
                authenticate();
                hibernateAlertDAO = new HibernateAlertDAO();
                hibernateAlertDAO.setSessionFactory((SessionFactory) applicationContext.getBean("sessionFactory"));
        }

        /**
         * Returns user with userId 1
         *
         * @return User
         */
        private User generateNewUser() {
                UserService userService = Context.getUserService();
                User user = userService.getUser(1);
                return user;
        }
```

Figure 4.5: HibernateAlertDAOTest.java (after fix, part 1)



```java
    /**
     * Create an Alert object
     *
     * @return Alert
     */
    private Alert genereateNewAlert() {
        Alert alert = new Alert(1);
        User user = generateNewUser();
        alert.setCreator(user);
        alert.setText("This is an alert");
        alert.setId(1);
        return alert;
    }

    /**
     * @see HibernateAlertDAO#saveAlert
     */
    @Test
    public void testSaveAlertShouldReturnNotNull() {
        Alert alert = genereateNewAlert();
        Alert savedAlert = hibernateAlertDAO.saveAlert(alert);
        assertNotNull(Integer.toString(savedAlert.getAlertId()));
    }

    /**
     * @see HibernateAlertDAO#getAlert
     */
    @Test
    public void testGetAlertShouldReturnAlert() {
        Alert alert = genereateNewAlert();
        hibernateAlertDAO.saveAlert(alert);
        alert = hibernateAlertDAO.getAlert(1);
        assertEquals(alert.getText(), "This is an alert");
    }
```

*Figure 4.6: HibernateAlertDAOTest.java (after fix, part 2)*



```java
    /**
     * @see HibernateAlertDAO#deleteAlert
     */
    @Test
    @Ignore
    public void testDeleteAlertShouldReturnNull() {
        Alert alert = genereateNewAlert();
        Alert save = hibernateAlertDAO.saveAlert(alert);
        hibernateAlertDAO.deleteAlert(save);
        assertNull(save.getId());

    }

    /**
     * @see HibernateAlertDAO#getAllAlerts
     */
    @Test
    public void testGetAllAlertsShouldReturnNotNull() {
        List<Alert> alerts = hibernateAlertDAO.getAllAlerts(true);
        assertNotNull(alerts);
    }

    /**
     * @see HibernateAlertDAO#getAlerts
     */
    @Test
    public void testGetAlertsShouldReturnNotNull() {
        User user = generateNewUser();
        List<Alert> alerts = hibernateAlertDAO.getAlerts(user, true, true);
        assertNotNull(alerts);
    }

}
```

*Figure 4.7: HibernateAlertDAOTest.java (after fix, part 3)*

## Instruction on how to apply new changes

1. Download the modified HibernateAlertDAOTest.java under TRUNK-5101 folder on github.
2. Swap with the file under:

   openmrs-core/api/src/test/java/org/openmrs/notification/db/hibernate/

   (make sure you have the latest version of openmrs-core)
3. Now HibernateAlertDAOTest is not empty anymore, and you can run it.

## 0.5 Technical Commentary

TRUNK-4762: Fixing this bug did not make much change in the design of the system since it was just a one-liner fix. Despite not changing much, we still feel we made a significant contribution since the UI looks cleaner now.

TRUNK-5101: Here we added a bunch of test cases in a previous empty test class. This is an improvement to the system since there should be as little untested code as possible.