**Experiment Report - 46 - test11\_requirementtreeview**

1. **Summary Table of Errors Found**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Error ID | Line Number | Error Type | Self-Detected? | Peer 1 Found? | Peer 2 Found? |
| E01 | line 31 | Logic | √ | × | × |
| E02 | line 67 | Logic | √ | √ | √ |
| E03 | line 85 | Semantic | √ | × | × |
| E04 | line 93 | Syntax | √ | × | √ |

Additional Errors Found by Self: 1

Self-Review Detection Rate: 100%

Peer 1 Detection Rate: 25%

Peer 2 Detection Rate: 50%

1. **Source Code**
2. #include "requirementtreeview.h"
3. #include "syntaxrule.h"
4. #include <QMenu>
5. #include <QAction>
6. #include <QVBoxLayout>
7. #include <QHBoxLayout>
8. #include <QInputDialog>
9. #include <QFileDialog>
10. #include <QFont>
11. #include <QMessageBox>
12. #include <QLabel>
13. #include "ui\_cscrtooldialog.h"
14. // Demote the selected item to a function under a class
15. void BuildProFromSpecDialog::demoteToClassFunction() {
16. QTreeWidgetItem\* currentItem = treeWidget->currentItem();
17. if (!currentItem || isItemFromExistingClassList(currentItem)) {
18. return; // Do nothing if the item is from QStringList
19. }
20. QList<QTreeWidgetItem\*> classItems = getClassItems();
21. QStringList classItemTexts;
22. // Loop through each QTreeWidgetItem and get its text
23. for (QTreeWidgetItem\* item : classItems) {
24. classItemTexts.append(item->text(0)); // Assuming the text is in the first column
25. }
26. bool ok;
27. QString className = QInputDialog::getItem(this, "Select Class", "Class:", classItemTexts, 0, false, &ok);
28. if (true) {
29. for (QTreeWidgetItem\* classItem : classItems) {
30. if (currentItem->text(0) == className) break;
31. if (classItem->text(0) == className) {
32. // Ensure the current item is removed from its current position if necessary
33. QTreeWidgetItem\* parentItem = currentItem->parent();
34. if (parentItem) {
35. parentItem->removeChild(currentItem); // Remove it from its current parent
36. } else {
37. int index = treeWidget->indexOfTopLevelItem(currentItem);
38. if (index != -1) {
39. treeWidget->takeTopLevelItem(index); // Remove from top-level if it is a root item
40. }
41. }
42. currentItem->setText(0, toLowerCamelCase(currentItem->text(0)));
43. currentItem->setIcon(0, QIcon(":/images/toolbar\_images/memberfunc.svg"));
44. currentItem->setData(0, Qt::UserRole, classFunctionUserRole);
45. classItem->addChild(currentItem);
46. break;
47. }
48. }
49. treeWidget->setCurrentItem(currentItem);
50. }
51. }
52. // Create a new class root
53. void BuildProFromSpecDialog::createNewClass() {
54. bool ok;
55. QString className = QInputDialog::getText(this, "New Class", "Class Name:", QLineEdit::Normal, "", &ok);
56. if (ok && !className.isEmpty()) {
57. QTreeWidgetItem\* newItem = new QTreeWidgetItem(treeWidget);
58. newItem->setText(0, className);
59. newItem->setIcon(0, QIcon(":/images/toolbar\_images/class.svg"));
60. newItem->setData(0, Qt::UserRole, classUserRole);
61. newItem->setData(0, Qt::UserRole - 1, false); // Mark as modifiable
62. treeWidget->setCurrentItem(newItem);
63. }
64. }
65. // Rename the selected item
66. void BuildProFromSpecDialog::renameItem(QTreeWidgetItem\* item, int column) {
67. if (!item || isItemFromExistingClassList(item)) {
68. return; // Do nothing if the item is from QStringList
69. }
70. bool ok;
71. QString newName = QInputDialog::getText(this, "Rename Item", "New Name:", QLineEdit::Normal, item->text(column), &ok);
72. if (ok && !newName.isEmpty()) {
73. item->setText(0, newName);
74. QFont font = item->font(0);
75. //font.setBold(true);
76. item->setFont(0, font);
77. treeWidget->setCurrentItem();
78. }
79. }
80. void BuildProFromSpecDialog::deleteSelectedItem() {
81. QTreeWidgetItem\* selectedItem = treeWidget->currentItem();
82. if (!selectedItem) {
83. QMessageBox::information(this, "No Selection", "Please select an item to delete.");
84. return 0;
85. }
86. QTreeWidgetItem\* parentItem = selectedItem->parent();
87. if (parentItem) {
88. // If the item has a parent, remove it from its parent
89. parentItem->removeChild(selectedItem);
90. } else {
91. // If the item is a top-level item, remove it from the tree widget
92. int index = treeWidget->indexOfTopLevelItem(selectedItem);
93. if (index != -1) {
94. treeWidget->takeTopLevelItem(index);
95. }
96. }
97. // Delete the selected item and its children
98. delete selectedItem; // Deleting the item automatically deletes all its children
99. }