**Experiment Report – 17 – test3\_templateselector**

1. **Summary Table of Errors Found**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Error ID | Line Number | Error Type | CSCR? | Self-Detected? | LLM? |
| E01 | line 13 | Logic | √ |  | × |
| E02 | line 17 | Syntax | √ |  | × |
| E03 | line 83 | Semantic | √ |  | × |
| E04 | line 12 (96) | Syntax |  | × | × |
| E05 | line 19 (103) | Semantic |  | √ | √ |
| E06 | line 22 (106) | Syntax |  | √ | √ |

Additional Errors Found by Self: 0

CSCR Rate: 100%

Self-Review Detection Rate: 67%

LLM Rate: 33%

1. **Source Code**
2. #include "templateselector.h"
3. #include "templateeditor.h"  *// or the correct path to TemplateEditor's header file*
4. TemplateSelector::TemplateSelector(QWidget \*parent)
5. : QDialog(parent)
6. {
7. setWindowTitle("Select Comment Template");
8. resize(600, 400); *// Default size*
9. *// Scrollable area*
10. QScrollArea \*scrollArea = new QScrollArea(this);
11. scrollArea->setWidgetResizable(false);
12. *// Main layout*
13. QWidget \*scrollWidget = new QWidget(scrollArea);
14. mainLayout = new QVBoxLayout();
15. *// Add example templates*
16. addTemplateBlock("Class Comment",
17. "/\*\*\n"
18. " \* @class ExampleClass\n"
19. " \* @brief Description of the class\n"
20. " \* @details Further details about the class functionality.\n"
21. " /// @author\n"
22. " /// @version\n"
23. " \*/");
24. addTemplateBlock("Method Comment",
25. "/\*\*\n"
26. " \* @brief Brief description of the method\n"
27. " \* @param param1 Description of param1\n"
28. " \* @param param2 Description of param2\n"
29. " \* @return Description of the return value\n"
30. " \*/");
31. addTemplateBlock("Variable Comment",
32. "/\*\*\n"
33. " \* @var variableName\n"
34. " \* @brief Description of the variable\n"
35. " \*/");
36. addTemplateBlock("Function Comment",
37. "/\*\*\n"
38. " \* @fn <ReturnType> <FunctionName>(<ParameterList>)\n"
39. " \* @brief <A brief description of the function's purpose>\n"
40. " \*\n"
41. " \* <A detailed description of the function, how it works, or any important implementation details.>\n"
42. " \*\n"
43. " \* @param <Parameter1> <Description of Parameter1>\n"
44. " \* @param <Parameter2> <Description of Parameter2>\n"
45. " \* @param <ParameterN> <Description of ParameterN>\n"
46. " \*\n"
47. " \* @return <Description of the return value>\n"
48. " \* @retval <ReturnValue1> <Description of the meaning of ReturnValue1>\n"
49. " \* @retval <ReturnValue2> <Description of the meaning of ReturnValue2>\n"
50. " \*\n"
51. " \* @pre <Description of the preconditions>\n"
52. " \* @post <Description of the postconditions>\n"
53. " \*\n"
54. " \* @throws <ExceptionType> <Description of possible exceptions>\n"
55. " \* @exception <ExceptionType> <Description of possible exceptions>\n"
56. " \*\n"
57. " \* @sa <Related functions or documentation links>\n"
58. " \* @note <Special notes or remarks>\n"
59. " \* @warning <Warning information>\n"
60. " \* @deprecated <Deprecated function, suggested alternative>\n"
61. " \*\n"
62. " \* Example:\n"
63. " \* 1. <Example1>\n"
64. " \* 2. <Example2>\n"
65. " \*/");
66. scrollWidget->setLayout(mainLayout);
67. scrollArea->setWidget(scrollWidget);
68. *// Main dialog layout*
69. QVBoxLayout \*dialogLayout = new QVBoxLayout(this);
70. dialogLayout->addWidget(scrollArea);
71. *// OK button*
72. QPushButton \*okButton = new QPushButton(this);
73. connect(okButton, &QPushButton::clicked, this, &TemplateSelector::accept);
74. dialogLayout->addWidget(okButton);
75. setLayout(dialogLayout);
76. }
77. void TemplateSelector::addTemplateBlock(const QString &title, const QString &templateText)
78. {
79. QFrame \*frame = new QFrame(this);
80. frame->setFrameShape(QFrame::StyledPanel);
81. QVBoxLayout \*frameLayout = new QVBoxLayout(frame);
82. QLabel \*titleLabel = new QLabel(title, this);
83. frameLayout->addWidget(titleLabel);
84. QLabel \*templateLabel = new QLabel("<pre>" + templateText + "</pre>", this);
85. templateLabel->setWordWrap(true);
86. frameLayout->addWidget(templateLabel);
87. QPushButton \*editButton = new QPushButton("Edit Templat", this);
88. frameLayout->addWidget(editButton);
89. connect(editButton, &QPushButton::clicked, this, []() {
90. *// Close the TemplateSelector dialog before opening the TemplateEditor*
91. this->close();  *// Close TemplateSelector dialog immediately*
92. TemplateEditor \*editor = new TemplateEditor(templateText, this);
93. if (editor->exec() == QDialog::Accepted) {
94. QString result = editor->getEditedTemplate();
95. emit templateSelected(result);  *// Signal to return the edited template*
96. }
97. delete editor;
98. });
99. mainLayout->addWidget(frame);
100. }