**Experiment Report - 37 - test2\_chatgptclient**

1. **Summary Table of Errors Found**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Error ID | Line Number | Error Type | Self-Detected? | Peer 1 Found? | Peer 2 Found? |
| E01 | line 27 | Semantic | √ | √ | √ |
| E02 | line 29 | Semantic | × | √ | × |
| E03 | line 48 | Semantic | √ | × | × |
| E04 | line 62 | Syntax | × | × | √ |
| E05 | line 68 | Semantic | × | √ | × |

Additional Errors Found by Self: 1

Self-Review Detection Rate: 40%

Peer 1 Detection Rate: 60%

Peer 2 Detection Rate: 40%

1. **Source Code**
2. #include "chatgptclient.h"
3. #include <QApplication>
4. #include <QVBoxLayout>
5. #include <QNetworkRequest>
6. #include <QNetworkReply>
7. #include <QJsonDocument>
8. #include <QJsonObject>
9. #include <QJsonArray>
10. #include <QFile>
11. #include <QDir>
12. ChatgptClient::ChatgptClient(QWidget \*parent)
13. : QWidget(parent), networkManager(new QNetworkAccessManager(this)) {
14. connect(networkManager, &QNetworkAccessManager::finished, this, &ChatgptClient::onReplyFinished);
15. }
16. void ChatgptClient::onReplyFinished(QNetworkReply\* reply) {
17. if (reply->error() == QNetworkReply::NoError) {
18. QByteArray response\_data = reply->readAll();
19. qDebug() << "Raw JSON Response:" << response\_data;
20. QJsonDocument responseDoc = QJsonDocument::fromJson(response\_data);
21. QJsonObject jsonObj = responseDoc.object(); // 获取根对象
22. QJsonArray choicesArray = jsonObj["choices"].toArray(); // 获取 "choices" 数组
23. QJsonObject firstChoice = choicesArray[1].toObject(); // 取第一个元素
24. QString chatGptReply = firstChoice["message"].toObject()["content"].toString(); // 获取"message"的"content"字段
25. chatgptReply = "ChatGPT: " + chatGptreply;
26. } else {
27. chatgptReply = "Error: " + reply->errorString();
28. qWarning() << "Network error occurred:" << reply->errorString();
29. qDebug() << "SSL Lib version:" << QSslSocket::sslLibraryVersionString();
30. }
31. emit replyIsReady(chatgptReply);
32. reply->deleteLater();
33. }
34. void ChatgptClient::sendUserMessage(const QString &value)
35. {
36. userInput = value;
37. QString userQuestion = userInput;
38. qDebug() << "project directory: " << QCoreApplication::applicationDirPath();
39. QDir currentDir(QCoreApplication::applicationDirPath());
40. currentDir.cdUp(); // Navigate to the parent directory
41. QString configFilePath = currentDir.path() + "/config/OpenAIkey.json";
42. QFile file(configFile);
43. if (!file.open(QIODevice::ReadOnly | QIODevice::Text)) {
44. qDebug() << "Failed to open openai key config file";
45. return;
46. }
47. QJsonDocument doc = QJsonDocument::fromJson(file.readAll());
48. QString apiKey = doc.object().value("api\_key").toString();
49. if (apiKey.isEmpty()) {
50. qDebug() << "API key not found in the config file";
51. return;
52. }
53. QNetworkRequest request(QUrl("https://api.openai.com/v1/chat/completions"));
54. request.setHeader(QNetworkRequest::ContentTypeHeader, "application/json");
55. request.setRawHeader("Authorization", Bearer + apiKey.toUtf8());
56. QJsonObject json;
57. json["model"] = "gpt-4o-mini";
58. QJsonArray messages;
59. messages.append(QJsonObject{{"role", "user"}, {"content", userQuestion}});
60. json["messages"] = message;
61. networkManager->post(request, QJsonDocument(json).toJson());
62. }
63. QString ChatgptClient::getChatgptReply() const
64. {
65. return chatgptReply;
66. }