МІНІСТЕРСТВО ОСВІТИ ТА НАУКИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ЛЬВІВСЬКА ПОЛІТЕХНІКА

Кафедра ЕОМ

АВТОМАТИЗОВАНЕ ПРОЕКТУВАННЯ КОМП'ЮТЕРНИХ СИСТЕМ

Task 5. Implement automated tests

Виконав:

Ст. гр. КІ-402

Середа Д.А.

Прийняв:

Федак П.Р.

Львів 2024

Мета: Впровадити автоматизовані тести для свого проекту згідно умов завдання.

Завдання

1. Implement or use existing test framework;

2. Create a set of automated tests;

3. Test report should contain number of all tests, passed tests, failed tests, coverage;

4. Coverage must be more than 80%

5. Required steps

Варіант:

19	tik-tac-toe 3x3	XML

Теоретичні відомості

Автоматизовані тести – це метод перевірки програмного забезпечення, при якому спеціально написані скрипти чи програми автоматично виконують тестування додатка, системи або її компонентів. Цей підхід дозволяє зменшити ручну працю, підвищити ефективність тестування та забезпечити повторюваність тестів.

Основна ідея автоматизованого тестування полягає в тому, щоб створити набір тестових сценаріїв, які можуть автоматично запускатися щоразу, коли вносяться зміни в код. Це допомагає швидко виявити помилки та гарантувати стабільність роботи програмного забезпечення.

Юніт-тести (unit tests) є основним методом тестування програмного забезпечення, спрямованим на перевірку окремих функцій або модулів програми в ізоляції від інших компонентів. Основна мета юніт-тестів — виявлення помилок на ранніх етапах розробки, що дозволяє швидко знаходити та виправляти помилки в логіці програми. Такі тести створюються зазвичай для кожної ключової функції чи методу програми, перевіряючи її правильність на основі заданих вхідних даних та очікуваних результатів. Юніт-тести сприяють збереженню високої якості коду, дозволяючи розробникам впевнено впроваджувати зміни, знаючи, що основний функціонал залишається стабільним.

У контексті Qt юніт-тести є частиною бібліотеки Qt Test, яка надає інструменти для створення та виконання автоматизованих тестів. Qt Test підтримує перевірку логіки коду (наприклад, класів та функцій) і роботи

графічного інтерфейсу (GUI). Зокрема, для GUI можна тестувати поведінку віджетів, взаємодію користувача та коректність відображення. Qt Test дозволяє створювати тести для перевірки поведінки додатка в різних сценаріях, включаючи крайні випадки. Це досягається за допомогою класів, таких як QTest, які підтримують емулювання дій користувача (натискання клавіш, кліки миші тощо), і засобів для перевірки результатів, наприклад, макросів QVERIFY чи QCOMPARE.

Юніт-тести в загальному, і Qt-тести зокрема, є важливими інструментами, які не лише підвищують якість програмного забезпечення, а й економлять час і ресурси в довгостроковій перспективі.

Деталі реалізації

Спочатку було налаштовано середовище для виконання автоматизованих тестів. У цьому проєкті використовувався вбудований у Qt фреймворк **Qt Test**, який надає інструменти для створення тестів. Тестування виконувалося через створення окремого тестового класу, що наслідує QObject. У цьому класі реалізовано методи з макросами Q_SLOT для підготовки, виконання тестів та завершення. Основні тести містили перевірку функціональності ключових компонентів програми.

```
********* Start testing of gameTest *********
Config: Using QtTest library 6.8.0, Qt 6.8.0 (x86_64-little_endian-llp64 s
PASS   : TestMainWindow::setupMenus()
PASS   : TestMainWindow::setupBoard()
PASS   : TestMainWindow::setupControls()
PASS   : TestMainWindow::handleCellClick()
FAIL   : TestMainWindow::makeAIMove()
PASS   : TestMainWindow::handleNewGame()
PASS   : TestMainWindow::handleModeChange()
PASS   : TestMainWindow::updateBoard()
PASS   : TestMainWindow::saveGame()
PASS   : TestMainWindow::loadGame()
PASS   : TestMainWindow::saveGameState()
PASS   : TestMainWindow::loadGameState()
PASS   : TestMainWindow::disableBoard()
PASS   : TestMainWindow::enableBoard()
FAIL   : TestMainWindow::checkGameEnd()
PASS   : TestGameLogic::reset()
PASS   : TestGameLogic::makeMove()
PASS   : TestGameLogic::checkWin()
PASS   : TestGameLogic::isBoardFull()
PASS   : TestGameLogic::isGameOver()
PASS   : TestGameLogic::getWinner()
PASS   : TestGameLogic::getBoardState()
PASS   : TestGameLogic::setBoardState()
FAIL   : TestGameLogic::minimax()
PASS   : TestGameLogic::checkWinForBoard()
PASS   : TestGameLogic::isBoardFullForBoard()
FAIL   : TestGameLogic::getBestMove()

Totals: 23 passed, 4 failed, 0 skipped, 0 blacklisted, 13ms
********* Finished testing of gameTest *********
```

Рис. 1. Результат виконання тестів у консолі.

У моїй програмі є 3 класи, (gamelogic, mainwindow, serialconnection), щоб знайти покриття спочатку додамо усі методи, сигнали та слоти. Отримуємо для gamelogic – 13, mainwindow – 15, а для serialconnection – 5;

Загальна кількість – 33

Непокриті методи – 6

Звідси отримуємо, що покриття становить 81.81%, що задовільняє завдання.

Висновок:

У ході виконання лабораторної роботи я успішно створив набір автоматизованих тестів, за допомогою фреймворка QT Test. Реалізував 27 тестових випадків, які тестують функціонал програми. Порахував загальне покриття тестами, що задовільняє виконання завдання. У результаті всі вимоги лабраторної роботи були виконані.

Список використаної літератури:

1. GitHub Docs, https://docs.github.com/
2. CSAD instructions for practical tasks and coursework