Редагуємо код за допомогою Copilot, щоб з’явилися тести:

Update my GitHub Actions ci.yml so that tests are actually built and executed. Inside every OS job (Linux, macOS, Windows), add the flag -DBUILD_TESTS=ON to the CMake configuration step. This ensures that GoogleTest and my unit_tests target are compiled. Also make sure that ctest --output-on-failure runs after the build, so the workflow fails if any test fails.

```name=.github/workflows/ci.yml
url=https://github.com/rarmax00/csad2526KI404HertsykMaksymAndriyovuch7/blob/main/.github/workflows/ci.yml
name: CI

on:
  push:
    branches:
      - '*develop*'
      - '*master*'
  pull_request:
    branches:
      - '*develop*'
      - '*master*'

jobs:
  cross_build:
    name: cross_build - ${{ matrix.os }}
    runs-on: ${{ matrix.os }}
    strategy:
      matrix:
        os: [ubuntu-latest, windows-latest, macos-latest]

    steps:
      - name: Checkout repository
        uses: actions/checkout@v4

      - name: Setup CMake
        uses: lukka/get-cmake@v3
```

```
    with:
      cmake-version: '3.24.3'

  - name: Build & Test (Linux/macOS)
    if: matrix.os != 'windows-latest'
    run: |
      set -e
      mkdir -p build
      cd build
      echo "Configuring with BUILD_TESTS=ON"
      cmake -DBUILD_TESTS=ON -DCMAKE_BUILD_TYPE=Release ..
      echo "Building"
      cmake --build . --config Release --parallel
      echo "Running tests"
      ctest --output-on-failure -C Release
    shell: bash

  - name: Build & Test (Windows)
    if: matrix.os == 'windows-latest'
    run: |
      REM Create build directory
      if not exist "build" mkdir build
      cd build
      echo Configuring with BUILD_TESTS=ON
      cmake -DBUILD_TESTS=ON -DCMAKE_BUILD_TYPE=Release ..
      echo Building
      cmake --build . --config Release -- /m
      echo Running tests
      ctest --output-on-failure -C Release
    shell: cmd

  - name: Upload build artifacts (deploy placeholder)
    uses: actions/upload-artifact@v4
    with:
```

```
      name: build-artifacts-${{ matrix.os }}
      path: |
        build/bin
        build\bin
```

I updated the CI workflow so every platform now configures CMake with -DBUILD_TESTS=ON and runs ctest --output-on-failure after the build. That ensures GoogleTest and the unit_tests target are built and that CI fails when any test fails.

Fix my CMakeLists.txt so that unit tests are always built and detected by CTest on all platforms. Currently no tests are found because BUILD_TESTS is OFF by default. I need you to remove the optional BUILD_TESTS flag and always enable GoogleTest, always call enable_testing(), and always add my test executable (tests.cpp). Make sure the target is called unit_tests, linked to math_ops and GTest::gtest_main, and registered via add_test(). Ensure that CTest can discover and run the tests in CI.

```name=CMakeLists.txt
url=https://github.com/rarmax00/csad2526KI404HertsykMaksymAndriyovuch7/blob/main/CMakeLists.txt
cmake_minimum_required(VERSION 3.14)

project(HelloWorld VERSION 1.0 LANGUAGES CXX)

# Enable CTest so tests are always discoverable
include(CTest)
enable_testing()

# Require C++17
set(CMAKE_CXX_STANDARD 17)
set(CMAKE_CXX_STANDARD_REQUIRED ON)
set(CMAKE_CXX_EXTENSIONS OFF)

# Create a small library for math operations so both the executable and tests can link to it
add_library(math_ops STATIC
  math_operations.cpp
)
```

```
target_include_directories(math_ops PUBLIC
  $<BUILD_INTERFACE:${CMAKE_SOURCE_DIR}>
)

# Create the main executable from main.cpp and link the math_ops library
add_executable(HelloWorld main.cpp)
target_link_libraries(HelloWorld PRIVATE math_ops)

# Prefer target-based properties
target_compile_features(HelloWorld PRIVATE cxx_std_17)
target_compile_features(math_ops PRIVATE cxx_std_17)

# Warning level / pedantic flags per compiler (helpful cross-platform defaults)
if (MSVC)
  target_compile_options(HelloWorld PRIVATE /W4 /permissive-)
  target_compile_options(math_ops PRIVATE /W4 /permissive-)
else ()
  target_compile_options(HelloWorld PRIVATE -Wall -Wextra -Wpedantic)
  target_compile_options(math_ops PRIVATE -Wall -Wextra -Wpedantic)
endif()

# Place binaries in a predictable folder inside the build directory (for all configs)
set_target_properties(HelloWorld math_ops PROPERTIES
  RUNTIME_OUTPUT_DIRECTORY "${CMAKE_BINARY_DIR}/bin"
  RUNTIME_OUTPUT_DIRECTORY_DEBUG "${CMAKE_BINARY_DIR}/bin"
  RUNTIME_OUTPUT_DIRECTORY_RELEASE "${CMAKE_BINARY_DIR}/bin"
  RUNTIME_OUTPUT_DIRECTORY_RELWITHDEBINFO "${CMAKE_BINARY_DIR}/bin"
  RUNTIME_OUTPUT_DIRECTORY_MINSIZEREL "${CMAKE_BINARY_DIR}/bin"
)

# Install rule for the produced executable
install(TARGETS HelloWorld
  RUNTIME DESTINATION bin
)
```

```
#
# GoogleTest: always fetch/build and always create/register unit_tests
#
include(FetchContent)

FetchContent_Declare(
  googletest
  URL https://github.com/google/googletest/archive/refs/heads/main.zip
)

# For MSVC: prefer shared CRT for gtest to match project settings if needed
if (MSVC)
  set(gtest_force_shared_crt ON CACHE BOOL "" FORCE)
endif()

FetchContent_MakeAvailable(googletest)

# Add the test executable and link to gtest_main and our math_ops library
# The test source file here is tests/unit_tests.cpp (keeps existing layout)
add_executable(unit_tests tests/unit_tests.cpp)
target_link_libraries(unit_tests PRIVATE GTest::gtest_main math_ops)
target_include_directories(unit_tests PRIVATE ${CMAKE_SOURCE_DIR})
target_compile_features(unit_tests PRIVATE cxx_std_17)

if (MSVC)
  target_compile_options(unit_tests PRIVATE /W4 /permissive-)
else ()
  target_compile_options(unit_tests PRIVATE -Wall -Wextra -Wpedantic)
endif()

# Ensure test binary lands in same predictable output folder for all configs
set_target_properties(unit_tests PROPERTIES
  RUNTIME_OUTPUT_DIRECTORY "${CMAKE_BINARY_DIR}/bin"
```

```
  RUNTIME_OUTPUT_DIRECTORY_DEBUG "${CMAKE_BINARY_DIR}/bin"
  RUNTIME_OUTPUT_DIRECTORY_RELEASE "${CMAKE_BINARY_DIR}/bin"
  RUNTIME_OUTPUT_DIRECTORY_RELWITHDEBINFO "${CMAKE_BINARY_DIR}/bin"
  RUNTIME_OUTPUT_DIRECTORY_MINSIZEREL "${CMAKE_BINARY_DIR}/bin"
)

# Register the test with CTest so it can be discovered and run in CI
add_test(NAME unit_tests COMMAND unit_tests)

# Provide a convenient 'check' target that runs CTest with output on failure.
add_custom_target(check
  COMMAND ${CMAKE_CTEST_COMMAND} --output-on-failure
  DEPENDS unit_tests
  WORKING_DIRECTORY ${CMAKE_BINARY_DIR}
)
```

I updated CMakeLists.txt to always enable testing and always fetch/build GoogleTest, create the unit_tests target (linked to math_ops and GTest::gtest_main), and register it with add_test so CTest discovers it reliably across platforms and generator types. I also forced test and application binaries into a predictable build/bin directory for all configurations so CI (multi-config or single-config) can find and run the test executable consistently. You can now run your CI workflow and CTest should discover and run the unit_tests target.

Update my GitHub Actions ci.yml so that tests are actually built and executed.
Inside every OS job (Linux, macOS, Windows), add the flag -DBUILD_TESTS=ON to the CMake configuration step.
This ensures that GoogleTest and my unit_tests target are compiled.
Also make sure that ctest --output-on-failure runs after the build, so the workflow fails if any test fails.

name=.github/workflows/ci.yml

```yaml
name: CI

on:
  push:
    branches:
      - '*develop*'
      - '*master*'
  pull_request:
    branches:
      - '*develop*'
      - '*master*'

jobs:
```

Fix my CMakeLists.txt so that unit tests are always built and detected by CTest on all platforms.
Currently no tests are found because BUILD_TESTS is OFF by default.
I need you to remove the optional BUILD_TESTS flag and always enable GoogleTest, always call enable_testing(), and always add my test executable (tests.cpp).
Make sure the target is called unit_tests, linked to math_ops and GTest::gtest_main, and registered via add_test().
Ensure that CTest can discover and run the tests in CI.

name=CMakeLists.txt

```
cmake_minimum_required(VERSION 3.14)

project(HelloWorld VERSION 1.0 LANGUAGES CXX)

# Enable CTest so tests are always discoverable
include(CTest)
enable_testing()

# Require C++17
set(CMAKE_CXX_STANDARD 17)
```

Ask anything

+ Add repositories, files, and spaces

GPT-5 mini

Установлюємо build-artifacts-windows-latest (ОС Windows), розпаковуємо, запускаємо, отримуємо результат Hello world: