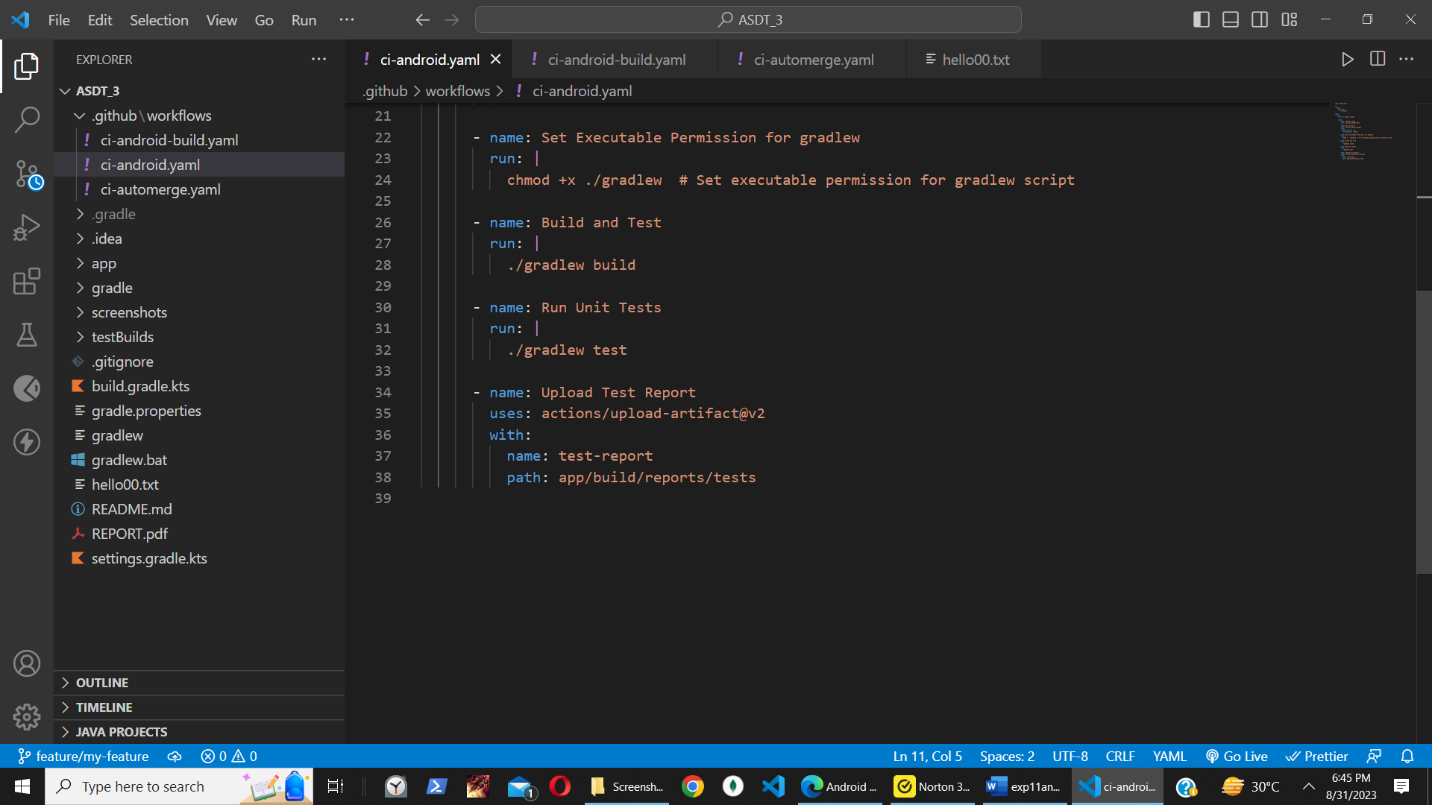
***EXP 11***

**First workflow : ANDROID CI WORKFLOW**

Code of the .yaml file :

A screenshot of a computer program

Description automatically generated

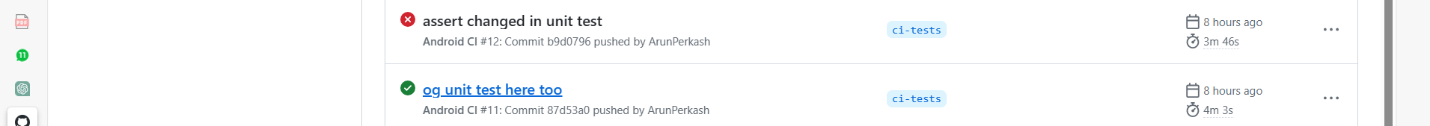


Github action:

A screenshot of a computer

Description automatically generated

Workflow that invokes success and failure:



Git commands for success and failure for workflow :

A black background with many dots

Description automatically generated

**Second workflow : CI-ANDROID-BUILD**

CODE:

A computer screen shot of a program

Description automatically generated

A screenshot of a computer program

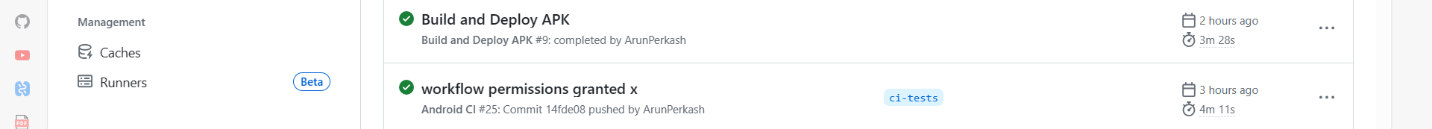
Description automatically generated

Github action:

A screenshot of a computer

Description automatically generated

Second workflow running after first successful workflow:



Git commands for the following on ci-tests branch ,

On push to ci-tests branch, we see the first workflow triggered followed by second:

A black background with a black square

Description automatically generated with medium confidence

**Assignment Completion Report :**

1. **Android CI Workflow:**

The "ci-android" workflow is triggered by pushes to the "ci-tests" branch. It serves as a cornerstone for maintaining code quality and identifying issues early in the development process. Here's a step-by-step breakdown:

*Event Trigger*: The workflow commences when changes are pushed to the dedicated "ci-tests" branch, signifying an intention to validate the changes through testing.

*Checkout Code:* This step employs the "actions/checkout" action to fetch the latest code from the repository, providing the environment with up-to-date code for testing.

*Set up JDK and Android Emulator*: By using the "actions/setup-java" action, the workflow establishes a Java Development Kit (JDK) environment. Subsequently, an Android emulator is initiated to create a simulated device for testing.

*Build and Test with Gradle*: Gradle, a powerful build tool, is employed to build the Android application. It compiles the source code, resolves dependencies, and produces an APK. Alongside, unit tests are executed to verify individual units of code, ensuring proper functionality.

*Upload Test Reports*: After running unit tests, the workflow uploads generated test reports to the repository. This aids in comprehensive test result analysis and facilitates bug identification and resolution.

*GitHub Status Checks:* GitHub status checks are updated to reflect the outcome of the workflow. This provides a visual indicator of whether the tests were successful or encountered issues.

1. **Workflow that invokes success** –

we added the correct unit test that worked correctly and resulted in a completion

**Workflow that invokes failure** –

we changed the assert in the unit test which led to the failure of the unit test and hence the entire workflow stopped and did not complete

1. **Build and Deploy APK Workflow:**

The "ci-android-build" workflow is set in motion upon the successful completion of the "ci-android" workflow. It focuses on building the Android APK, renaming it, and organizing it within the repository. Here's a detailed look:

Event Trigger: This workflow is initiated only when the "ci-android" workflow successfully completes its tasks, ensuring that the code has passed the required tests.

Checkout Code: Similar to the previous workflow, the "actions/checkout" action is utilized to fetch the latest code from the repository.

Set up JDK: The "actions/setup-java" action configures the environment with JDK 17, a critical component for building the Android APK.

Set Executable Permission: A shell command grants executable permission to the Gradle wrapper script, allowing it to be run as part of the build process.

Build Android APK: The Gradle wrapper is executed with the "assembleDebug" task, resulting in the construction of the Android APK in debug mode.

Rename APK with Timestamp: The APK generated in the previous step is renamed to include a timestamp, appending the date and time of creation. This is useful for version tracking and identification.

Create Repository Folder: A new folder named "testBuilds" is created at the root of the repository. This folder is intended to store the APK and maintain a clean organizational structure.

Move APK to Folder: The renamed APK is moved into the "testBuilds" folder for storage. This step contributes to a structured repository layout.

Git Operations: Git commands are executed to facilitate the version control process. The workflow configures the author's identity, stages the new APK, commits the changes with a descriptive message, and finally pushes the changes back to the repository.