**Practical Exercise 04 - Working with Third-Party Custom Actions**

**Exercise Description**

**In this practical exercise, our goal is to explore how we can use third-party custom actions to perform tasks without having to define them from scratch.**

In order to achieve that, we will leverage a React application that we will scaffold with the help of the create-react-app utility. Check the tips section for the specific command to scaffold your React application. Here are the instructions for the exercise:

1. Generate a React application:
   1. Create a new folder named 04-using-actions at the root of the repository.
   2. Using a terminal, cd into this directory and scaffold a React application inside a react-app directory. You can either create the directory yourself or let the create-react-app utility do it for you.
   3. Once the React setup is done, you should see a success message.
   4. Take a few moments to inspect the files and get familiar with the application folder structure.
2. Create the first version of the workflow:
   1. Create a file named 04-using-actions.yaml under the .github/workflows folder at the root of your repository.
   2. Name the workflow 04 - Using Actions.
   3. Add the following triggers to your workflow:
      1. push
   4. Add a single job named build to the workflow. The job should contain two steps:
      1. The first one, named Checkout Code, should checkout the repository code into the current working directory.
      2. The second one, named Printing Folders, should simply print the folder structure after the checkout command.
   5. Commit the changes and push the code.
   6. Take a few moments to inspect the output of the workflow run.
3. Extend the workflow to setup node and install the dependencies of the React application
   1. Remove the Printing Folders step.
   2. Add a new step after the Checkout Code step. This new step should be named Setup Node and setup node using the 20.x version.
   3. Add a new step after the Setup Node step. This new step should be named Install Dependencies install the dependencies of our React application by running the npm ci command inside the React application folder. You can either cd into the directory before running the npm ci command, or you can pass the working directory by adding a working-directory: 04-using-actions/react-app option to the step (as a sibling key to name and run).
   4. Commit the changes and push the code.
   5. Take a few moments to inspect the output of the workflow run.
4. Extend the workflow to execute the automated tests from the React application:
   1. Add a new step after the Install Dependencies step. This new step should be named Run Unit Tests and it should execute the automated tests by running the npm run test command inside the React application folder. You can either cd into the directory before running the npm run test command, or you can pass the working directory by adding a working-directory: 04-using-actions/react-app option to the step (as a sibling key to name and run).
   2. Commit the changes and push the code.
   3. Take a few moments to inspect the output of the workflow run.
5. Change the workflow triggers to contain only workflow\_dispatch to prevent this workflow from running with every push and pollute the list of workflow runs.

**Tips**

**Scaffolding a React application with create-react-app**

To generate a React application with a single command, you can leverage the create-react-app utility. In order to do that, simply run the following inside the 04-using-actions folder: npx create-react-app --template typescript react-app.

**Using third-party actions in GitHub Actions Workflows**

The syntax to leverage a third-party action is very simple. Instead of using the run key and defining a shell script, you should use the uses key and pass the name and the version of the action you wish to use. Here is an example of the syntax:

1. steps:
2. - name: Using the Checkout Action
3. uses: actions/checkout@v4

**Useful third-party actions for this exercise**

1. actions/checkout@v4: used to checkout the repository code into the working directory of the workflow run. Without this, we cannot work with the code from our repository.
2. actions/setup-node@v4: used to setup node with a specific version, as well as any other necessary dependencies. To provide a specific version of Node to be setup, you can use the following syntax:
   1. steps:
   2. - name: Setup Node
   3. uses: actions/setup-node@v4
   4. with:
   5. node-version: '20.x'