TESTING IN GIT HUB:

To test your code in GitHub, you can use **GitHub Actions**, which is a built-in Continuous Integration/Continuous Deployment (CI/CD) tool. GitHub Actions allows you to automatically run your tests every time you push changes to your repository or create a pull request.

Here’s how you can set up GitHub Actions to test your code:

**1. Create a GitHub Actions Workflow**

1. In your project directory, create the following folder structure:

.github/workflows/

1. Inside the workflows folder, create a file named:
2. test.yml.
3. **2. Define the Workflow in test.yml**
4. Add the following content to the test.yml file. This workflow will:
5. Set up Python.
6. Install your dependencies.
7. Run your unittest tests.

name: Python Tests

on:

  push:

    branches:

      - main

  pull\_request:

    branches:

      - main

jobs:

  test:

    runs-on: ubuntu-latest

    steps:

    - name: Checkout code

      uses: actions/checkout@v3

    - name: Set up Python

      uses: actions/setup-python@v4

      with:

        python-version: '3.9'  # Use the Python version your project requires

    - name: Install dependencies

      run: |

        python -m venv venv

        source venv/bin/activate

        pip install -r requirements.txt

    - name: Run tests

      run: |

        source venv/bin/activate

        python -m unittest discover -s tests

**3. Push the Workflow to GitHub**

1. Add the workflow file to your Git repository:

git add .github/workflows/test.yml

git commit -m "Add GitHub Actions workflow for testing"

git push

1. Once pushed, GitHub will automatically detect the workflow and run it.

**4. View the Test Results**

1. Go to your GitHub repository in your browser.
2. Click on the **Actions** tab.
3. You should see a workflow named **Python Tests** running. Click on it to view the details.
4. If the tests pass, you’ll see a green checkmark. If they fail, you’ll see a red "X" with details about the failure.

**5. Debugging Test Failures**

If any tests fail:

1. Check the logs in the **Actions** tab to see which test failed and why.
2. Fix the issue locally, commit the changes, and push them to GitHub to rerun the tests.

**6. Optional: Add a Status Badge**

You can add a status badge to your repository’s README file to show the status of your tests.

1. Go to the **Actions** tab in your repository.
2. Click on the workflow (e.g., **Python Tests**).
3. Click on the **...** menu in the top-right corner and select **Create status badge**.
4. Copy the markdown code and paste it into your README.md file:

**7. Test Locally Before Pushing**

Before pushing to GitHub, you can run your tests locally to ensure they pass:

**8. Automate Database Setup (Optional)**

If your tests depend on a MySQL database, you can automate the database setup in GitHub Actions by adding steps to:

1. Install MySQL.
2. Set up the database schema using your SQL file (Final airline\_reissues.sql).

Here’s an example of how to include MySQL setup in your workflow:

TESTING FAILED IN GITHUB:

The test in your GitHub Actions workflow failed because the requirements.txt file could not be found. Specifically, the error message is:

ERROR: Could not open requirements file: [Errno 2] No such file or directory: 'requirements.txt'

This issue typically arises when the workflow tries to install Python dependencies, but the requirements.txt file is missing from the repository or not at the expected path.

**Steps to Fix:**

1. **Ensure requirements.txt Exists:**
   * Verify that the requirements.txt file is in the root directory of your repository or the specified path in the workflow.
2. **Update the Workflow File:**
   * If the requirements.txt file is in a different location, update the path in your workflow file (.github/workflows/test.yml) under the step where dependencies are installed. For example:

- name: Install dependencies

run: pip install -r path/to/requirements.txt

**Commit the File:**

* If the file is missing, create a requirements.txt in the repository, listing the Python dependcies required for your project, such as:

Code:

* mysql-connector-python
* pytest
  + Commit the file and push the changes.

1. **Re-run the Workflow:**
   * After fixing the issue, re-run the workflow in GitHub Actions to check if the problem is resolved.

Let me know if you'd like help reviewing the workflow file or creating the requirements.txt!

TOKEN FOR GITHUB:

ghp\_ArWveP6YQYc6uSYU7I1XD7wtWXHi4B3Tdpiy