Assignment #4

Deliverables:

You are required to submit a SINGLE Zip file that has the following deliverables are:

1. Your IPYNB scripts
2. All of your source code and output
3. Output report that has your assignment run saved in OUTPUT.pdf
4. Video recording of 15-20 minutes as a demo for the run of your assignment using Panopto

Important Notes:

- Do NOT communicate or share your assignment with others
- Do NOT share your personal laptop with your classmates
- You MUST preserve the same directory structure that is provided in this assignment

High-Level Requirements:

1. The assignment will be completed in 3 phases.
2. All your deliverables will be based on work you have done on GitHub.
3. Phase-I is due on 3/26/2024 by 11:59pm
4. Phase-II is due on 3/29/2024 by 11:59pm
5. Phase-III is due on 4/7/2024 by 11:59pm

Phase-I Requirements:

1. You must create your GitHub account ID as instructed in the tutorial.

- For a student with the name John Doe with A number A12345678.
- His id will be JSP24SCM78D.
- In this ID J is the first character of his first name.
- SP24SCM will be common for each student.
- 78 is the last two numbers of his A number.
- D is the first character of his last name.

  Once the TA receives all of your GitHub accounts you will be added as Collaborators to the master repository. For this, the TA will be sending out a mail to get all your created GitHub accounts. Use your hawk id while registering for your GitHub accounts. Do not use newly created email ids just for the assignment as GitHub may consider these ID's as Bots and remove them. You will be proceeding to the next step only after the TA sends out a mail confirming that you have been added as a collaborator.

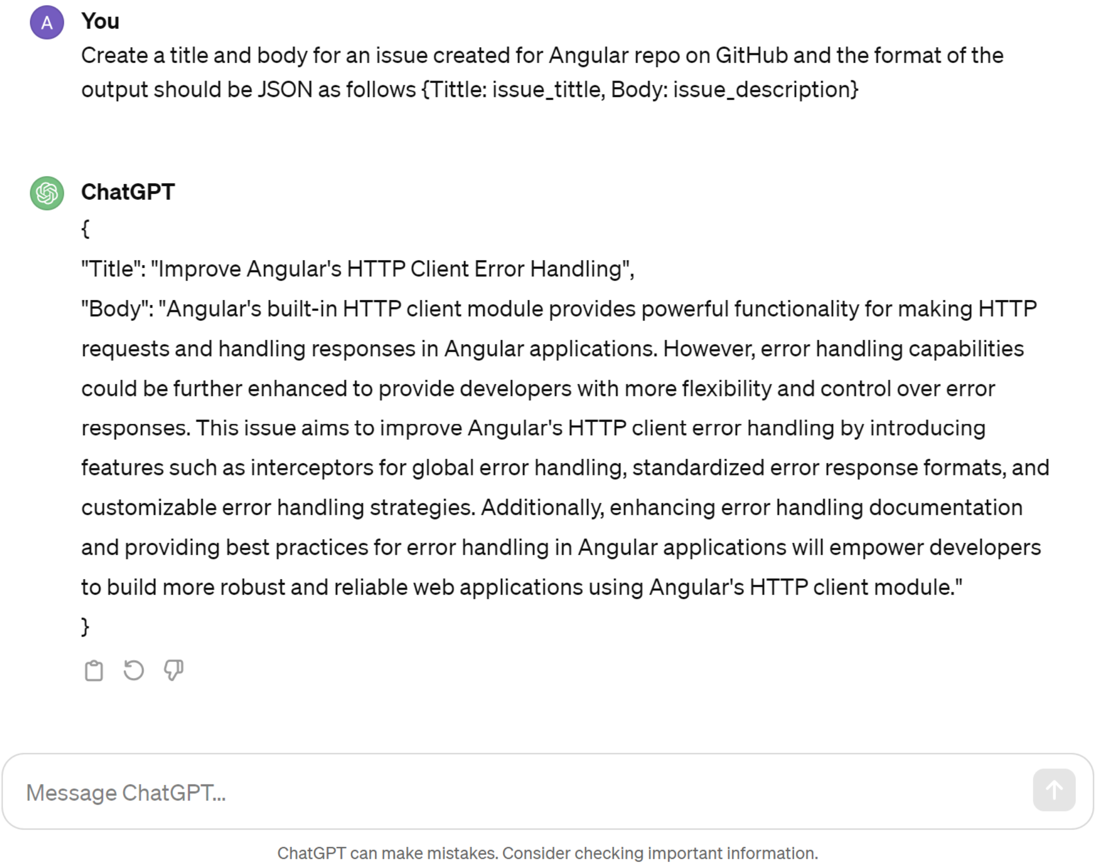
2. Next you will be forking the repository SCM587SP24 from SPM587SP24 GitHub organization to the GitHub account that you have created. This repository can be opened with the following URL: https://github.com/SPM587SP24/SCM587SP24.git

3. Now using the URL given above you will login to the repository listed above. You will go to the issues tab and create a new issue.

4. Use OpenAI ChatGPT 3.5 and create the issue **title** and **body**, and then copy & paste those entries to GitHub issues tab. Use the following instructions to select the Repo name for the issue you are creating using OpenAI ChatGPT 3.5:

   Based on the first letter of your **First** name, select the repo name for the issue as follows:
   - A-F : Use Angular as the Repo Name
   - G-L : Use d3.js as the Repo Name
   - M-R : Use React as the Repo Name
   - S-Z : Use OpenAI-cookbook as the Repo Name

See below an example of a prompt to generate the issue **title** and **body**:



You

Create a title and body for an issue created for Angular repo on GitHub and the format of the output should be JSON as follows {Tittle: issue_tittle, Body: issue_description}

ChatGPT

{
"Title": "Improve Angular's HTTP Client Error Handling",
"Body": "Angular's built-in HTTP client module provides powerful functionality for making HTTP requests and handling responses in Angular applications. However, error handling capabilities could be further enhanced to provide developers with more flexibility and control over error responses. This issue aims to improve Angular's HTTP client error handling by introducing features such as interceptors for global error handling, standardized error response formats, and customizable error handling strategies. Additionally, enhancing error handling documentation and providing best practices for error handling in Angular applications will empower developers to build more robust and reliable web applications using Angular's HTTP client module."
}

ChatGPT can make mistakes. Consider checking important information.

5. This issue will be **labeled** using the labeling scheme listed below.

6. **<u>Labeling Scheme</u>**: Your Labeling Scheme follows the **key:value** format "**LabelName:*LabelValue***" for every label
7. **<u>Types of Labels</u>**: There are different types of labels, the following is the list of labels used for issues listed that you would use based on the **5th digit of your A-Number**:
    - i. **OriginationPhase**:
        1. **If the 5th digit is odd number:** use one of the values{Coding, Testing, Documentation, Field}
        2. **If the 5th digit is even number:** use one of the values{Requirements, Design}
    - ii. **DetectionPhase**:
        1. **If the 5th digit is odd number:** use one of the values{Coding}

        2. **If the 5th digit is even number:** use one of the values{Design}
    - iii. **Priority**:
        1. **If the 5th digit is odd number:** use one of the values {Major, Low}
        2. **If the 5th digit is even number:** use one of the values{Medium, Critical}
    - iv. **Status**:
        1. **If the 5th digit is odd number:** use one of the values {Rejected, Completed}
        2. **If the first digit is even number:** use one of the values{pendingReview,Approved, inProgress, }
    - v. **Category**:
        1. **If the 5th digit is odd number:** use one of the values {Enhancement, Inquiry}
        2. **If the 5th digit is even number:** use one of the values{Bug}

8. You will now clone the repository that you have forked into your account on to your local machine. This repository will contain files a.java, b.java, c.java, …, z.java. You will open the file that has the first character of your first name. Once you open the file you will enter your GitHub account id in a new line and save it.

    Example – For John Doe with GitHub account JSP24SCM78D, he will open j.java and add his GitHub account username in a new line and save it. So j.java will contain JSP24SCM78D.

9. Now you will do a commit and then push this repository into your GitHub account. While making the commit you must add your Issue number in the commit message in the following format.

    Example – git commit –m “Fixes #<IssueNumber>. <GitHub username>Adding first name.”

    Say for John Doe the issue he has created is Issue #24. His commit message will be as follows.

    ```
    git commit –m “Fixes #24. JSP24SCM78DAdding first name”
    ```

10. Phase 1 Report

- ✓ Create a report with 2 screenshots. See instructions for taking screenshots in the next section.
- ✓ Follow this naming convention for your report for phase 1. **Phase1_Report_<GitHub UserId>.pdf**.
  For example, Phase1_Report_JSP24SCM78D.pdf
- ✓ The report that you have created will be added to the report folder in the repository and submitted. This means that for the phase I report, you will be adding the report document in that ***Phase1_Report*** folder and committing it.
- ✓ Make sure that the report for each phase must be committed and pushed to your repository in Phase1_Report folder BEFORE you initiate a pull request.

11. Once that is done you will create a pull request to the master repository so that your changes can be merged onto the master repository. You will just create the pull request. It will be the TA who will review your changes and merge it. Your pull request title should also be the exact same as your commit message.

This will complete your Phase I. *No Blackboard submission is needed for Phase I*.

Once your changes are all accepted in the master repository you will receive an email to commence Phase II. Then you will be following the Phase II instructions.

Do not start Phase II till you have received the email to proceed with Phase II from the TA.

Phase-II Requirements:

1. You will open the master repository from your GitHub account and create a new issue.
2. Use OpenAI ChatGPT 3.5 and create the issue **title** and **body,** and then copy & paste those entries to GitHub issues tab. Use the following instructions to select the Repo name for the issue you are creating using OpenAI ChatGPT 3.5:

   Based on the first letter of your **Last** name, select the repo name for the issue as follows:
   - A-F : Use Angular as the Repo Name
   - G-L : Use d3.js as the Repo Name
   - M-R : Use React as the Repo Name
   - S-Z : Use OpenAI-cookbook as the Repo Name

3. This issue will be **labeled** using the labeling scheme listed below.
4. **Labeling Scheme**: Your Labeling Scheme follows the **key:value** format "**LabelName:*LabelValue***" for every label
5. **Types of Labels**: There are different types of labels, the following is the list of labels used for issues listed that you would use based on the **last digit of your A-Number**:
   i. **OriginationPhase**:
      1. **If the last digit is odd number:** use one of the values{Requirements, Design}
      2. **If the last digit is even number:** use one of the values{Coding, Testing, Documentation, Field}

   ii. **DetectionPhase**:
      1. **If the last digit is odd number:** use one of the values{Testing}
      2. **If the last digit is even number:** use one of the values{Field}

   iii. **Priority**:
      1. **If the last digit is odd number:** use one of the values {Critical,Major}
      2. **If the last digit is even number:** use one of the values{High, Low, Medium}

   iv. **Status**:
      1. **If the last digit is odd number:** use one of the values {Approved, Rejected, Completed}
      2. **If the last digit is even number:** use one of the values {inProgress, pendingReview }

   v. **Category**:

    1. **If the last digit is odd number:** use one of the values {Bug, Enhancement}
    2. **If the last digit is even number:** use one of the values{Inquiry}

6. Now you go to the forked repository in your account and update the repository. This is done using the following command.

   ```
   git remote add upstream <Master Repository Git URL>
   git fetch upstream
   git rebase upstream/master
   ```

7. Update your local clone to the latest in your repository. Make sure you have the latest changes in your local clone to avoid any conflicts.

8. Now you will open the file with the first character of your last name. Then you will be adding your GitHub account username in a new line

   Example – John Doe will be opening the file d.java and add his GitHub account username which is JSP24SCM78D in a new line.

9. Now you will do a commit and then push this repository into your GitHub account. While making the commit you are to add your Issue number in the commit message in the following format.

   Example – git commit –m “Fixes #<IssueNumber>. <GitHub username>Adding last name”

   Say for John Doe the issue he has created is Issue #65. His commit message will be as follows.

   ```
   git commit –m “Fixes #65. JSP24SCM78DAdding last name”
   ```

10. Phase 2 Report

✓ Create a report with 2 screenshots. See instructions for taking screenshots in the next section.

✓ Follow this naming convention for your report for phase 2. **Phase2_Report_<GitHub UserId>.pdf**.
For example, Phase2_Report_JSP24SCM78D.pdf

- ✓ The report that you have created will be added to the report folder in the repository and submitted. This means that for the phase II report, you will be adding the report document in that ***Phase2_Report*** folder and committing it.
- ✓ Make sure that the report for each phase must be committed and pushed to your repository BEFORE you initiate a pull request.

11. Once that is done you will create a pull request to the master repository so that your changes can be merged onto the master repository. You will just create the pull request. It will be the TA who will review your changes and merge it. Your pull request title should also be the exact same as your commit message.

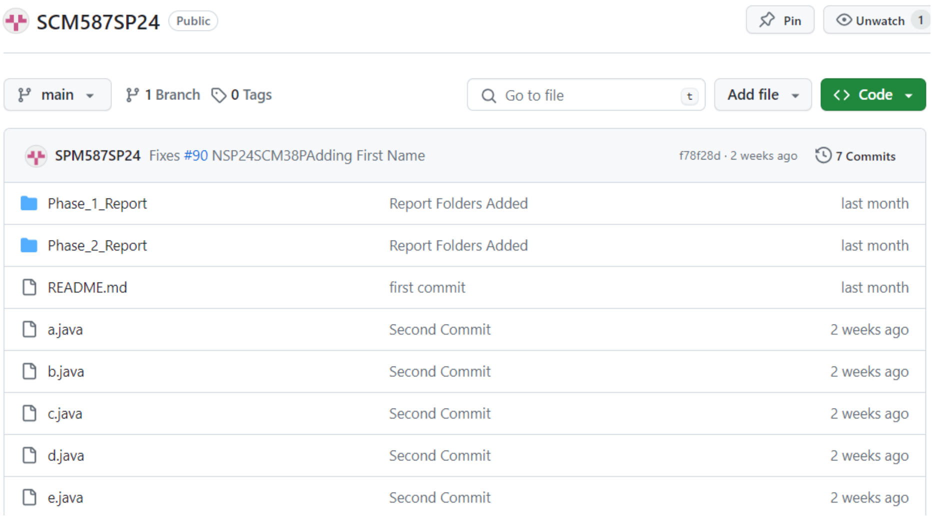
This will complete your Phase II<u>. *No Blackboard submission is needed for Phase II*</u>.

Once your changes are all accepted in the master repository you will receive an email to commence Phase III. Then you will be following the Phase III instructions.

<u>**Do not start Phase II till you have received the email to proceed with Phase II from the TA.**</u>

Screenshot Instructions - Phase I &Phase-II:

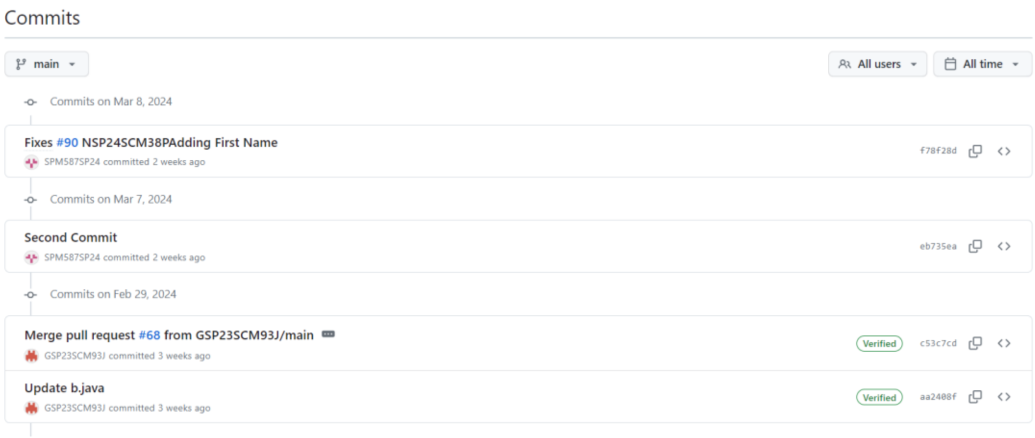
1. Login to your GitHub account and open the repository that you have committed to. Click on the commits Tab.



SPM587SP24 Fixes #90 NSP24SCM38PAdding First Name f78f28d · 2 weeks ago

Phase_1_Report	Report Folders Added	last month
Phase_2_Report	Report Folders Added	last month
README.md	first commit	last month
a.java	Second Commit	2 weeks ago
b.java	Second Commit	2 weeks ago
c.java	Second Commit	2 weeks ago
d.java	Second Commit	2 weeks ago
e.java	Second Commit	2 weeks ago

2. You will get the following page. You should take a screenshot of the below page. Click on the commit that you have just made to expand it.



Commits on Mar 8, 2024

Fixes #90 NSP24SCM38PAdding First Name f78f28d
SPM587SP24 committed 2 weeks ago

Commits on Mar 7, 2024

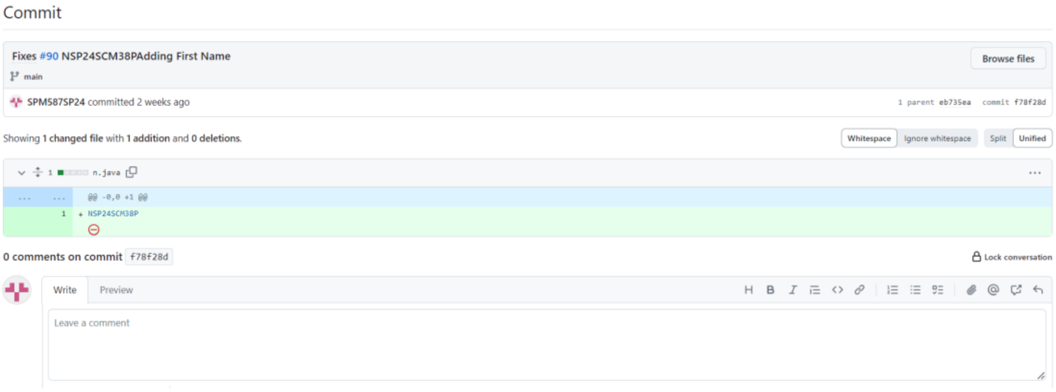
Second Commit eb735ea
SPM587SP24 committed 2 weeks ago

Commits on Feb 29, 2024

Merge pull request #68 from GSP23SCM93J/main Verified c53c7cd
GSP23SCM93J committed 3 weeks ago

Update b.java Verified aa2408f
GSP23SCM93J committed 3 weeks ago

3. This will expand and show you the individual commit. You must take a screenshot of this page as well.



Fixes #90 NSP24SCM38PAdding First Name

main

SPM587SP24 committed 2 weeks ago

1 parent eb735ea commit f78f28d

Showing 1 changed file with 1 addition and 0 deletions.

n.java

```
@@ -0,0 +1 @@
1 + NSP24SCM38P
```

0 comments on commit f78f28d

Leave a comment

Phase-III Requirements:

Here is the URL to guide you on how to generate your GITHUB_TOKEN https://help.github.com/articles/creating-an-access-token-for-command-line-use/

After you create your GitHub personal access tokens from https://github.com/settings/tokens, update and run the IPYNB script ***Importing_Issues_from_GitHub.ipynb*** to import the issues from the REPO.

1. Update and run the modified ***Charting_imported_issues.ipynb*** script to plot the following charts:
   - Requirement #1: Plot in Bar Chart the total number of issues closed every day for every Origination Phase
   - Requirement #2: Plot in Bar Chart the total number of issues created for every Phase based on their Status
2. Your updated **GitHub_Issues_ES_Docker_OpenAI** files that have
   - Requirement #1: You must create/publish public Docker images on **DockerHub** for **ElasticSearch image** with the data/embeddings using the same process demonstrated in the tutorials for Push and Pull of Docker images.
   - Requirement #2: [PUSH] Reuse/Modify the provided IPYNB scripts to create and store the index and embeddings for repo:SPM587SP24/SCM587SP24 using Docker/ElasticSearch8.12
   - Requirement #3: [Pull] Reuse/Modify the provided IPYNB scripts to pull and query the index/embeddings for repo:SPM587SP24/SCM587SP24 using Docker/ElasticSearch8.12 in order to perform a sematic search to find the top 5 similar issues that are in the index.

- Requirement #4: [Pull] Reuse/Modify the provided IPYNB scripts to pull and query the index/embeddings for repo:SPM587SP24/SCM587SP24 using Docker/ElasticSearch8.12 in order to perform a sematic search to find the **top 5 similar Rendering React issues that are created in our repo**.

- Requirement #5: [Pull] Reuse/Modify the provided IPYNB scripts to pull and query the index/embeddings for repo:SPM587SP24/SCM587SP24 using Docker/ElasticSearch8.12 in order to perform a sematic search to find the **top 5 similar Rendering React issues that are created in our repo between 3/27/2024 and 3/29/2024**.