CCS3313 - Advanced Software Design

Version Controlling a Class Diagram Using Git

Library Management System

Report

Group No: 16

1. 22UG1-0323 | W.G. Kasun Chamika De Mel

2. 22UG1-0472 | K.G. Pasindu Kavishka

3. 22UG2-0034 | N.N.K. Arachchi

4. 22UG2-0032 | D.B. Aveesha Umali

5. 22UG2-0588 | P.D. Lakshan Janith

Overview:

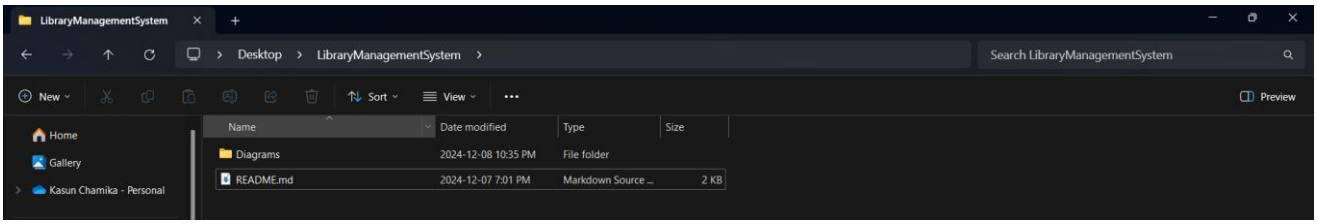
Version control systems are essential in modern software development, providing an organized way to track changes and enable seamless collaboration among team members working on the same project. This assignment focused on leveraging Git, a distributed version control system, to manage and version-control the class diagram for a Library Management System.

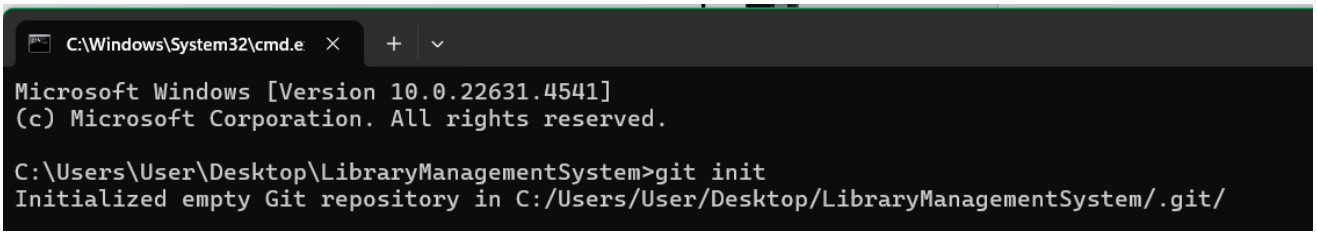
The class diagram represents the key entities and their relationships within the system, making it a critical component of the design process. As part of this assignment, we carried out various practical activities, including initializing a Git repository, collaborating on updates to the class diagram, resolving merge conflicts, and consolidating changes into a unified version.

These tasks not only enhanced our understanding of Git commands and workflows but also improved our skills in conflict resolution and team collaboration. By actively engaging in these activities, we gained valuable experience in maintaining version control in a team-oriented development environment, ensuring that all contributions were properly tracked and integrated into the project.

⇨ ***What we did***

Setting Up the Project: We started by creating a folder 'LibraryManagementSystem' and initializing it as a Git repository. The class diagram was saved in a 'Diagrams' folder as a png, and we wrote a brief description of the project in a README.md file. Everything was committed and pushed to GitHub so the team could work collaboratively.





```
C:\Users\User\Desktop\LibraryManagementSystem>git add .
warning: in the working copy of 'Diagrams/Untitled Diagram.drawio.xml', LF will be replaced by CRLF the next time Git to
uches it

C:\Users\User\Desktop\LibraryManagementSystem>git commit -m "Initial commit with class diagram and README"
[master (root-commit) 999bed2] Initial commit with class diagram and README
 3 files changed, 223 insertions(+)
 create mode 100644 Diagrams/22UG1-0323_ClassDiagram.png
 create mode 100644 Diagrams/Untitled Diagram.drawio.xml
 create mode 100644 README.md
```

- Logged into Gitllub and created a remote repository.
- Linked the local repository to the remote using git remote add origin'.
- Pushed the initial code to the remote repository.

```
C:\Users\User\Desktop\LibraryManagementSystem>git remote add origin https://github.com/KasunChamikaDeMel/LibraryManageme
ntSystem.git

C:\Users\User\Desktop\LibraryManagementSystem>git branch -M main

C:\Users\User\Desktop\LibraryManagementSystem>git push -u origin main
info: please complete authentication in your browser...
Enumerating objects: 6, done.
Counting objects: 100% (6/6), done.
Delta compression using up to 16 threads
Compressing objects: 100% (6/6), done.
Writing objects: 100% (6/6), 33.97 KiB | 11.32 MiB/s, done.
Total 6 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/KasunChamikaDeMel/LibraryManagementSystem.git
 * [new branch]      main -> main
branch 'main' set up to track 'origin/main'.
```

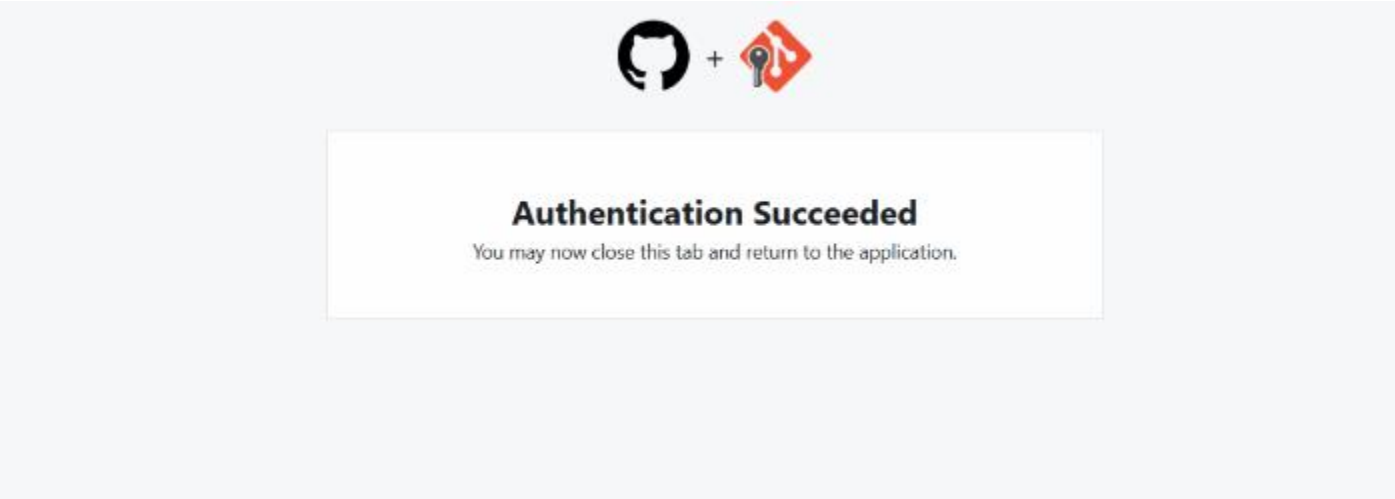
Team Contributions:

Each member cloned the repository and added their name to the Contributors section in the README. We also worked on improving the class diagram by adding new relationships and making it more accurate for a library management system.

Working with Branches:

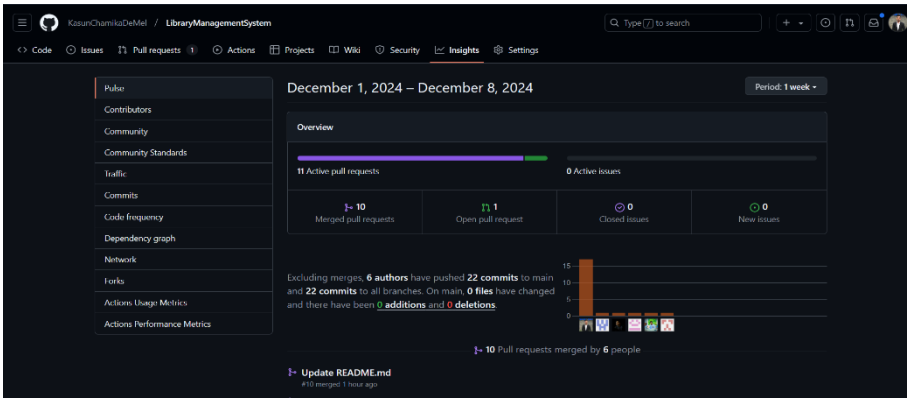
Before making any changes, we created a new branch called 'update-README' to avoid disrupting the main branch. Once the updates were made, the changes were committed and pushed to the branch. A pull request was opened and reviewed the changes before merging them into the main branch.

```
C:\Users\User\Desktop\LibraryManagementSystem>git push origin update-README
Switched to a new branch 'update-README'
```



Handling Conflicts:

To practice resolving conflicts, two of us edited the same part of the README file on different branches. When we tried to merge the branches, Git flagged a conflict. We resolved it by editing the conflicting file locally, saving the changes, and completing the merge.



Final Touches: Once everything was merged, we made sure the class diagram and README file were up-to-date and pulled the latest changes so everyone had the final version.

- ⇨ ***Challenges We Faced***

Learning Git: Some of us were new to Git, so it took time to understand commands like branching, merging, and resolving conflicts.

Merge Conflicts: Resolving conflicts was tricky at first, but with some practice, we got the hang of it.

Team Coordination: Making sure everyone worked on their own branch and didn't overwrite each other's changes required clear communication and planning.

- ⇨ ***What We Learned***

Using Git Effectively: We now understand how to set up and use a Git repository, work with branches, and push changes to a remote repository.

Pull Requests and Code Reviews: Pull requests are a great way to review changes before merging, and they helped us maintain a smooth workflow.

Resolving Conflicts: Resolving merge conflicts showed us how important it is to communicate and keep our work organized.

Best Practices for Collaboration: We learned to write clear commit messages, pull changes regularly, and stay organized to avoid unnecessary conflicts.

- ⇨ ***Conclusion:***

This project was a great hands-on experience in working with version control systems like Git and GitHub. We now feel more confident collaborating on code, managing changes, and resolving conflicts. These skills will help us in future projects!