Karnataka Law Society's

GOGTE INSTITUTE OF TECHNOLOGY

Udyambag Belagavi -590008

Karnataka, India.

A Course Activity Report on

GITLAB – A Agile Development Tool

Submitted for the requirements of 7th semester B.E. in CSE

for **"Agile Software Development"**

Submitted by

NAME	**USN**
Mr. Danish Sajan	2GI20CS035
Mr. Nimisha G J	2GI20CS074
Mr. Puneet Joshi	2GI20CS102
Mr. Rohan Shinde	2GI20CS110

Under the guidance of

Prof. Raghvendra Y Jadhav

Assistant Professor, Dept. of CS

Academic Year 2023-2024 (Odd semester)

Karnataka Law Society's

GOGTE INSTITUTE OF TECHNOLOGY

Udyambag Belagavi -590008

Karnataka, India.

Department of Computer Science and Engineering

Certificate

This is to certify that the Course Activity work titled **"GITLAB – A Agile Development Tool"** carried out by **Mr. Danish Sajan, Mr. Nimisha G J , Mr. Puneet Joshi ,Mr. Rohan Shinde** bearing **USNs: 2GI20CS035, 2GI20CS074, 2GI20CS102, 2GI20CS110**for **"Agile Software Development" (18CS753)** course is submitted in partial fulfilment of the requirements for 7th semester B.E. in **COMPUTER SCIENCE AND ENGINEERING,** Visvesvaraya Technological University, Belagavi. It is certified that all corrections/ suggestions indicated have been incorporated in the report. The course project report has been approved as it satisfies the academic requirements prescribed for the said degree.

Date:

Place: Belagavi

Signature of Guide

Prof. Raghvendra Y Jadhav

Dept. of CS KLS
Gogte Institute of
Technology, Belagavi

Karnataka Law Society's
GOGTE INSTITUTE OF TECHNOLOGY
Udyambag Belagavi -590008

Academic Year 2023-24(Odd Semester)

Semester: 7

Course: Agile Software Development (18CS753)

<u>Rubrics for evaluation of Course Project</u>

S.No	**Project / Seminar**	**Name→**	**Mr . Danish Sajan**	**Mr. Nimisha G J**	**Mr. Puneet Joshi**	**Mr. Rohan Shinde**
		USN →	**2GI20CS035**	**2GI20CS074**	**2GI20CS102**	**2GI20CS110**
		Max. Marks				
1	Relevance of the project and its objectives	02				
3	Demonstration / Presentation	03				
4	Q and A	02				
5	Project Report	03				
	Total	**10**				

<u>**INDEX**</u>

What is a Version Control System?

A version control system (VCS), also known as a source control system or revision control system, is a software tool or system that manages changes to files, documents, or source code over time. It enables multiple individuals or teams to collaborate on a project by tracking and recording modifications, allowing for the coordination of work and the ability to revert to previous states of a project. Version control systems are widely used in software development but are also valuable in other fields where document management is critical.

Key features of a version control system include:

- Change Tracking: VCS tracks changes to files, recording who made each change and when it was made. This is valuable for accountability and collaboration.
- Version History: VCS maintains a history of all changes, allowing users to review, compare, and revert to previous versions of files or code.
- Concurrency: It enables multiple individuals to work on the same project simultaneously, managing conflicts and merging changes.
- Branching and Merging: VCS systems provide the ability to create branches, or parallel lines of development, and merge them back into the main project. This is especially useful for developing new features or isolating bug fixes.
- Conflict Resolution: When multiple users make conflicting changes to the same file, the VCS helps identify and resolve those conflicts.
- Backup and Recovery: VCS serves as a backup mechanism, ensuring that project data is not lost due to human error or technical issues.

There are two main types of version control systems:

Centralized Version Control System (CVCS): In a CVCS, there is a single central repository where all project files and version history are stored. Users check out files to work on them and then check them back in when done. Examples of CVCS include CVS and Subversion (SVN).

Distributed Version Control System (DVCS): In a DVCS, every user has a complete copy of the repository, including the full history. Users can work independently and synchronize changes with other repositories. Popular DVCS systems include Git, Mercurial, and Bazaar.

Git is one of the most widely used version control systems, known for its distributed nature, speed, and flexibility. It is the VCS behind popular platforms like GitHub, GitLab, and Bitbucket, and it has applications far beyond software development, including managing configurations, documentation, and more.

Version control systems are essential tools for managing and collaborating on software projects, ensuring data integrity, and maintaining a record of changes and decisions over time. They are valuable for teams of all sizes, from small development teams to large organizations working on complex projects.

What is Gitlab?

Git is a distributed version control system (DVCS) that revolutionized the way people manage and collaborate on software development projects. Created by Linus Torvalds in 2005, Git is known for its speed, flexibility, and the ability to handle projects of all sizes. It is widely used not only in software development but also in other domains where version control and collaborative work are essential. To understand Git thoroughly, let's explore its key concepts and how it functions.

GitLab is a web-based platform that provides comprehensive end-to-end DevOps lifecycle management. It is primarily known for its version control capabilities, but it offers a wide range of tools for software development and collaboration. To understand GitLab deeply, let's explore its key features and components.

Key Features and Components of GitLab:

- Version Control (Git Repository): GitLab is built around Git, a distributed version control system, and it provides a web-based interface for managing Git repositories. Users can create, clone, and manage Git repositories for their projects.
- Web-based Interface: GitLab offers an intuitive web-based user interface for creating and managing projects, branches, commits, and more. Users can access GitLab through their web browsers, making it easy to collaborate.
- Issue Tracking: GitLab includes an integrated issue tracking system for managing tasks, bugs, and feature requests. It allows for the creation, assignment, and tracking of issues within a project.
- Continuous Integration/Continuous Deployment (CI/CD): GitLab provides a built-in CI/CD pipeline system that automates the building, testing, and deployment of code changes. This helps ensure code quality and accelerate the development process.
- Container Registry: GitLab features a container registry where you can store and manage Docker containers. This is useful for containerized application deployment.
- Code Review: GitLab offers a code review system that allows team members to collaborate and provide feedback on code changes before they are merged into the project.
- Merge Requests: Merge requests are used to propose changes from one branch to another. They often include code changes, discussions, and a review process before merging.
- Access Control: GitLab provides fine-grained access control, enabling project owners to define who can view, edit, or administer a project. This is crucial for security and collaboration.
- Wiki and Documentation: Projects in GitLab can include wikis and documentation, making it easy to create and maintain project-specific documentation.
- Security Scanning: GitLab offers built-in security features, including static and dynamic application security testing (SAST and DAST) to identify vulnerabilities in your code.
- Auditing and Compliance: GitLab provides auditing capabilities to track changes and user activities. It also supports compliance requirements with features like merge request approvals.
- Integration: GitLab integrates with various third-party tools, services, and external repositories. It can integrate with cloud providers, chat services, issue trackers, and more.
- Self-hosted and SaaS Options: GitLab can be hosted on your own servers or used as a Software-as-a-Service (SaaS) platform, providing flexibility in deployment.

Software development process in Gitlab:-

1. Create a Project: Users create a new project in GitLab. This project will host the source code, documentation, and related assets.

2. Version Control: Developers push their code changes to GitLab repositories using Git. GitLab tracks these changes, creating a complete version history.

3. Issue Tracking: Issues are created to track tasks, bugs, and feature requests. They can be linked to code changes for reference.

4. Code Review: Developers create merge requests to propose code changes. These changes undergo code review, discussion, and automated testing.

5. CI/CD Pipeline: GitLab's CI/CD pipelines automatically build, test, and deploy code changes. Developers receive feedback on the success or failure of these pipelines.

6. Merge and Deploy: Code changes that pass the review and CI/CD pipeline are merged into the main branch and deployed to production.

7. Security Scanning: GitLab's built-in security features scan code for vulnerabilities, helping to identify and mitigate security risks.

8. Documentation: Project documentation and wikis are maintained in GitLab to provide information for developers and users.

Commonly used GIT commands:-

Initialization:

git init: Initializes a new Git repository.

Cloning:

git clone <repository_url>: Creates a local copy of a remote Git repository on your computer.

Configuration:

git config --global user.name "Your Name": Sets your name for Git commits.

git config --global user.email "youremail@example.com": Sets your email for Git commits.

Adding and Committing Changes:

git add <filename>: Stages changes for the next commit.

git commit -m "Your commit message": Creates a snapshot of the project with a description of changes.

git commit -a -m "Your commit message": Stages and commits all changes in tracked files.

Branching and Merging:

git branch: Lists local branches.

git branch <branch_name>: Creates a new branch.

git checkout <branch_name>: Switches to a different branch.

git merge <branch_name>: Merges changes from one branch into another.

git branch -d <branch_name>: Deletes a local branch.

Remote Repository Interaction:

git remote -v: Lists remote repositories.

git remote add <remote_name> <remote_url>: Adds a new remote repository.

git pull <remote_name> <branch_name>: Fetches and merges changes from a remote repository.

git push <remote_name> <branch_name>: Pushes your local changes to a remote repository.

Checking the Status and History:

git status: Shows the status of your working directory.

git log: Displays the commit history.

git log --oneline: Shows a concise commit history.

git diff: Shows the differences between the working directory and the most recent commit.

git diff <source_branch> <target_branch>: Compares the changes between two branches.

Tagging:

git tag: Lists all tags.

git tag -a <tag_name> -m "Tag description": Creates an annotated tag for a specific commit.

git push --tags: Pushes tags to a remote repository.

Stashing:

git stash: Temporarily saves changes that are not ready to be committed.

git stash apply: Applies the most recent stash.

git stash list: Lists all stashes.

Reverting:

git reset --hard <commit_hash>: Resets the branch to a specific commit, discarding all changes after that commit.

git revert <commit_hash>: Creates a new commit that undoes the changes introduced by a specific commit.

Submodules:

git submodule add <repository_url> <path>: Adds a submodule to a Git repository.

git submodule init: Initializes submodules.

git submodule update: Updates submodules to the latest commit.

<u>**Menus in Gitlab:**</u>

Dashboard:

The dashboard provides an overview of your GitLab activity. It displays an activity feed of recent events, such as commits, issues, and merge requests related to the projects you're involved in.

Projects:

The "Projects" menu is where you can manage and access all your GitLab projects. You can create new projects, search for existing ones, and access project-specific settings.

Groups:

Groups in GitLab allow you to organize projects and users. You can create groups, invite users to join them, and manage group settings and access control.

Explore:

The "Explore" option provides access to publicly available projects, groups, and user profiles on GitLab. It's a way to discover and explore open-source projects and communities.

Issues:

The "Issues" menu is where you manage project-related tasks, bugs, and feature requests. You can create, assign, label, and track the status of issues.

Merge Requests:

"Merge Requests" are used to propose code changes and collaborate on them. This menu provides access to create, review, and merge merge requests.

CI/CD/Settings:

In the context of a project, "CI/CD/Settings" gives access to GitLab's Continuous Integration and Continuous Deployment settings and configurations. It's where you define and automate your build and deployment pipelines.

Wiki:

The "Wiki" menu provides access to project-specific documentation and notes. You can create and edit pages to document project-related information.

Security & Compliance:

The "Security & Compliance" menu contains GitLab's built-in security and compliance features, such as static and dynamic application security testing (SAST and DAST).

Repository:

In the "Repository" menu, you can view, browse, and manage the source code of your GitLab project. It provides a web-based interface for code navigation, browsing commits, and examining code history.

Members:

"Members" is where you manage project collaborators and permissions. You can invite team members, assign roles, and configure access control for your project.

Operations:

The "Operations" menu contains tools for managing various project operations, including environments, Kubernetes clusters, and serverless functions.

Settings:

The "Settings" menu, specific to a project, allows you to configure project-level settings, such as general project information, repository settings, and integration options.

Analytics:

In the "Analytics" menu, you'll find tools that provide insights into project activities, repository statistics, and more. This is valuable for tracking project progress and performance.

DevOps:

"DevOps" is where you access GitLab's integrated DevOps tools, including the CI/CD pipeline, monitoring features, and error tracking. This menu is essential for managing the development and deployment processes.

Package & Registries:

The "Package & Registries" menu is where you can manage package and container registries. You can publish, access, and manage Docker images, NPM packages, and other artifacts.

Issue Boards:

Issue boards are visual tools for managing tasks and workflows. The "Issue Boards" menu provides access to create and manage these boards.

Integrations:

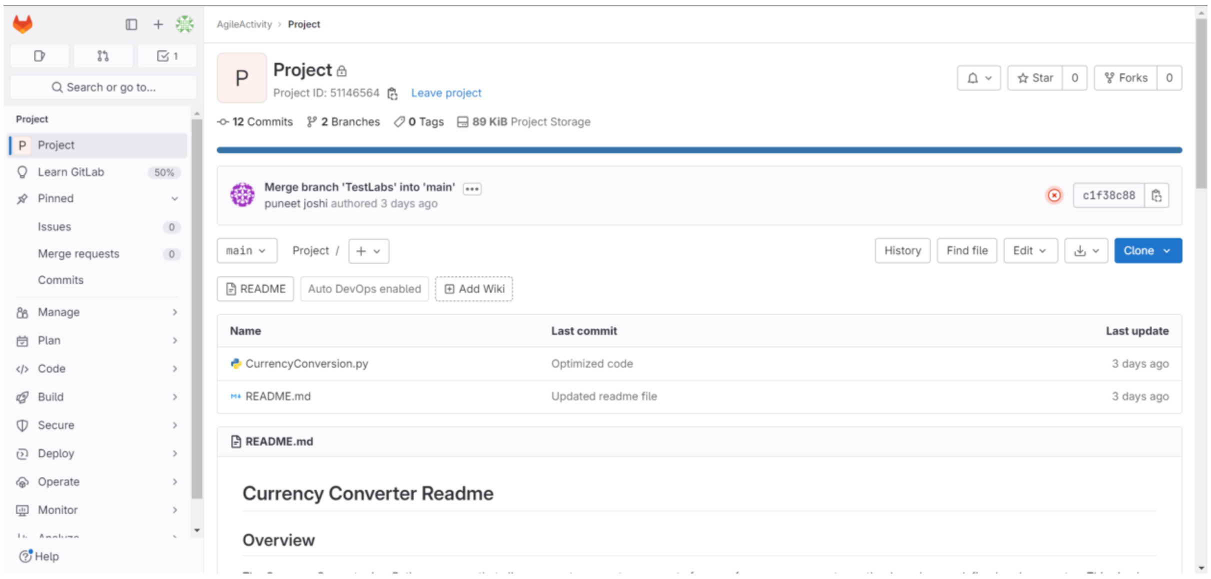
In the "Integrations" menu, you can set up various integrations with third-party services and tools. This includes webhooks and services that connect GitLab with other applications.

Help:

The "Help" menu typically provides links to GitLab's documentation, community forum, and support resources. It's a valuable resource for finding answers to questions and getting support.

Your Avatar/Profile:

Clicking on your avatar or profile picture allows you to access your user profile settings, preferences, and log out of GitLab.



Project

Project ID: 51146564

12 Commits 2 Branches 0 Tags 89 KiB Project Storage

Merge branch 'TestLabs' into 'main'
puneet joshi authored 3 days ago
c1f38c88

main Project /

Name	Last commit	Last update
CurrencyConversion.py	Optimized code	3 days ago
README.md	Updated readme file	3 days ago

README.md

Currency Converter Readme

Overview

Teminalogies used in development cycle:-

Merge Event:

A merge event in GitLab refers to the process of combining changes from one branch into another, typically from a feature branch into the main branch (often called "master" or "main"). This is a critical part of the version control process and is a common event in software development. Merge events are typically associated with merge requests (also known as pull requests in other version control systems), which are requests to merge changes from one branch into another.

When a merge event occurs, it signifies that the changes made in a feature branch are integrated into the main branch, making them a part of the project's official codebase. Merge events help maintain a clean and organized version history by incorporating only the completed and reviewed changes into the main branch. GitLab provides tools to facilitate and track the merge process.

During a merge event, developers may review the code changes, discuss potential conflicts, and ensure that the new code aligns with project goals and coding standards. Merge events help maintain code quality and collaboration in a team by providing a structured process for integrating changes.

Issue Event:

An issue event in GitLab represents the creation, management, and tracking of tasks, bugs, and feature requests related to a software development project. Issues serve as a central hub for managing work items, organizing tasks, and facilitating communication among team members.

Issues can cover a wide range of topics, such as fixing a bug, implementing a new feature, or addressing technical debt. When a team member encounters a problem or identifies an improvement opportunity, they can create an issue to document and track the work that needs to be done.

GitLab's issue tracking system allows team members to assign issues to individuals, apply labels for categorization, set due dates, and maintain a history of comments and discussions related to each issue. This enables effective project management, collaboration, and the prioritization of work items.

Issue events help in planning, organizing, and prioritizing tasks throughout the software development lifecycle. They also serve as a reference point for discussions and decisions regarding project progress.

Branch Event:

A branch event in GitLab is associated with the creation, management, and tracking of branches within a Git repository. Branches are a fundamental concept in version control systems and are used to isolate different lines of development within a project.

Developers typically create branches to work on specific features, bug fixes, or experiments without affecting the main codebase. Branches allow parallel development and help avoid conflicts between multiple developers working on the same project. Branches can be created from existing branches, making it easy to branch off from the main branch or other feature branches.

GitLab provides tools for managing branches, including creating, renaming, and deleting branches. Each branch has its own commit history, making it easy to track changes related to a specific feature or task. Branches are a critical part of a Git workflow, as they facilitate the isolation and organization of code changes.

Branch events reflect the process of creating, merging, and retiring branches as part of the software development cycle. Effective branch management is crucial for maintaining code quality and ensuring a smooth development process.

Push Event:

A push event in GitLab occurs when a developer pushes their local code changes to a remote Git repository. When developers work on their local machines, they make changes to the codebase, commit those changes, and then push those commits to a remote repository. This action ensures that the code changes are synchronized with the shared project repository, making them accessible to other team members.

Push events are significant because they enable collaboration and version control. They trigger actions such as continuous integration (CI) pipelines, which automatically build and test the code changes. Push events are tracked, and GitLab provides a record of the commits associated with each push.

In GitLab, push events often play a role in the continuous integration and continuous deployment (CI/CD) process. A successful push event initiates a CI/CD pipeline, which automates the process of building, testing, and deploying the code changes to various environments.

In summary, these GitLab events are fundamental to the software development process:

Merge events integrate changes into the main codebase.

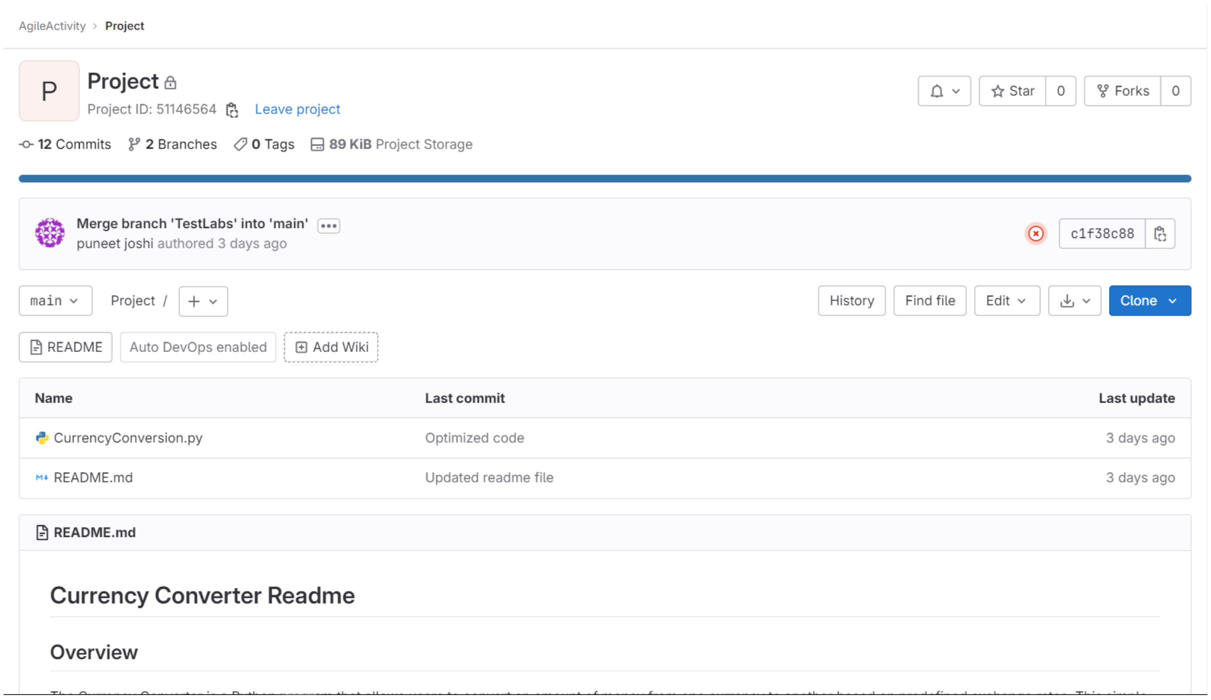
Issue events manage and track tasks, bugs, and feature requests.

Branch events facilitate parallel development and code isolation.

Push events synchronize code changes with the project repository, often triggering CI/CD pipelines.

Screenshots of project, Activity, members tabs:-

Project Panel:


AgileActivity › Project

Project
Project ID: 51146564 Leave project

12 Commits 2 Branches 0 Tags 89 KiB Project Storage

Merge branch 'TestLabs' into 'main'
puneet joshi authored 3 days ago
c1f38c88

main / Project / History Find file Edit Clone

README Auto DevOps enabled Add Wiki

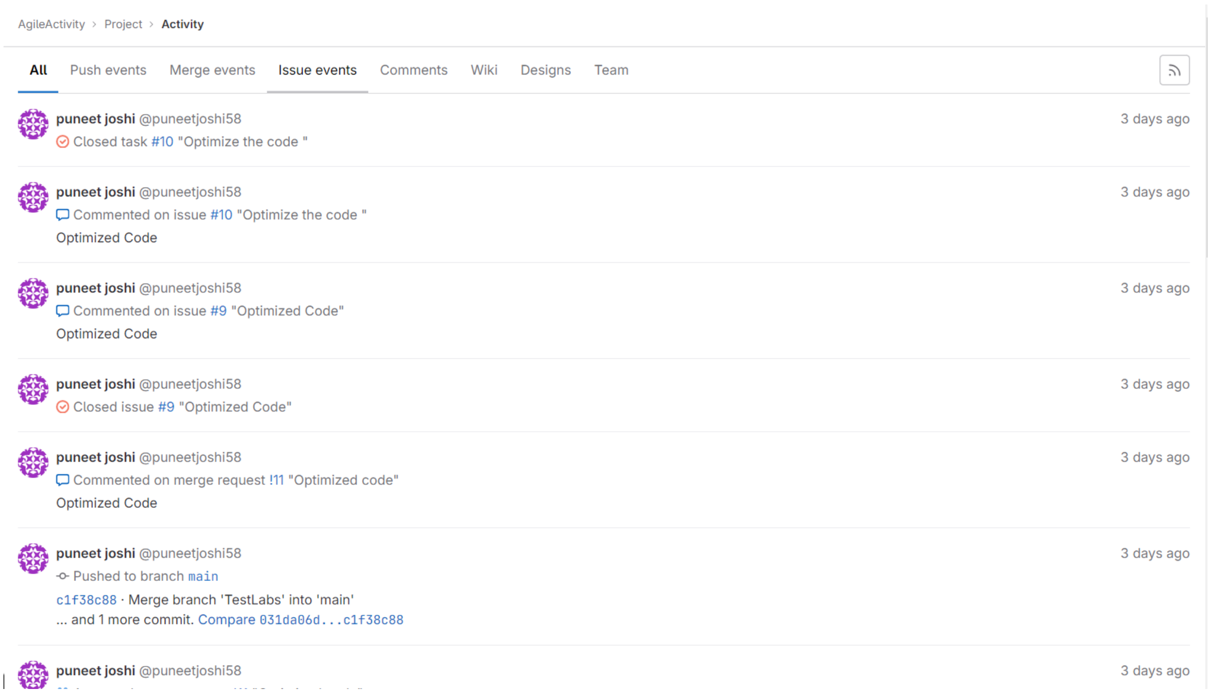
Name	Last commit	Last update
CurrencyConversion.py	Optimized code	3 days ago
README.md	Updated readme file	3 days ago

README.md

Currency Converter Readme

Overview

Activity Panel:


AgileActivity › Project › Activity

All Push events Merge events Issue events Comments Wiki Designs Team

puneet joshi @puneetjoshi58 3 days ago
Closed task #10 "Optimize the code "

puneet joshi @puneetjoshi58 3 days ago
Commented on issue #10 "Optimize the code "
Optimized Code

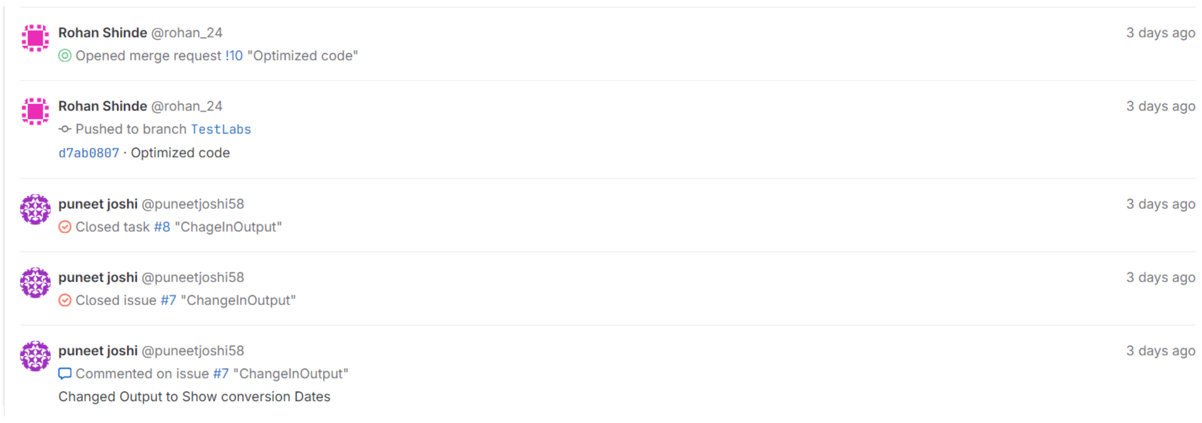
puneet joshi @puneetjoshi58 3 days ago
Commented on issue #9 "Optimized Code"
Optimized Code

puneet joshi @puneetjoshi58 3 days ago
Closed issue #9 "Optimized Code"

puneet joshi @puneetjoshi58 3 days ago
Commented on merge request !11 "Optimized code"
Optimized Code

puneet joshi @puneetjoshi58 3 days ago
Pushed to branch main
c1f38c88 · Merge branch 'TestLabs' into 'main'
... and 1 more commit. Compare 031da06d...c1f38c88

puneet joshi @puneetjoshi58 3 days ago



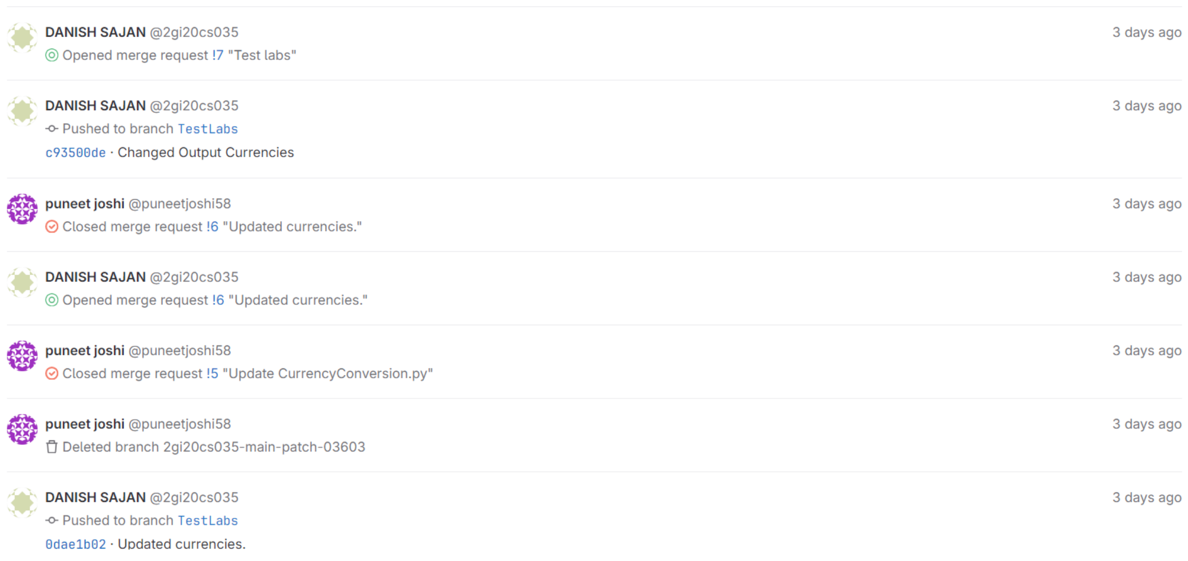
Rohan Shinde @rohan_24 3 days ago
Opened merge request !10 "Optimized code"

Rohan Shinde @rohan_24 3 days ago
Pushed to branch `TestLabs`
`d7ab0807` · Optimized code

puneet joshi @puneetjoshi58 3 days ago
Closed task #8 "ChagelnOutput"

puneet joshi @puneetjoshi58 3 days ago
Closed issue #7 "ChangelnOutput"

puneet joshi @puneetjoshi58 3 days ago
Commented on issue #7 "ChangelnOutput"
Changed Output to Show conversion Dates



DANISH SAJAN @2gi20cs035 3 days ago
Opened merge request !7 "Test labs"

DANISH SAJAN @2gi20cs035 3 days ago
Pushed to branch `TestLabs`
`c93500de` · Changed Output Currencies

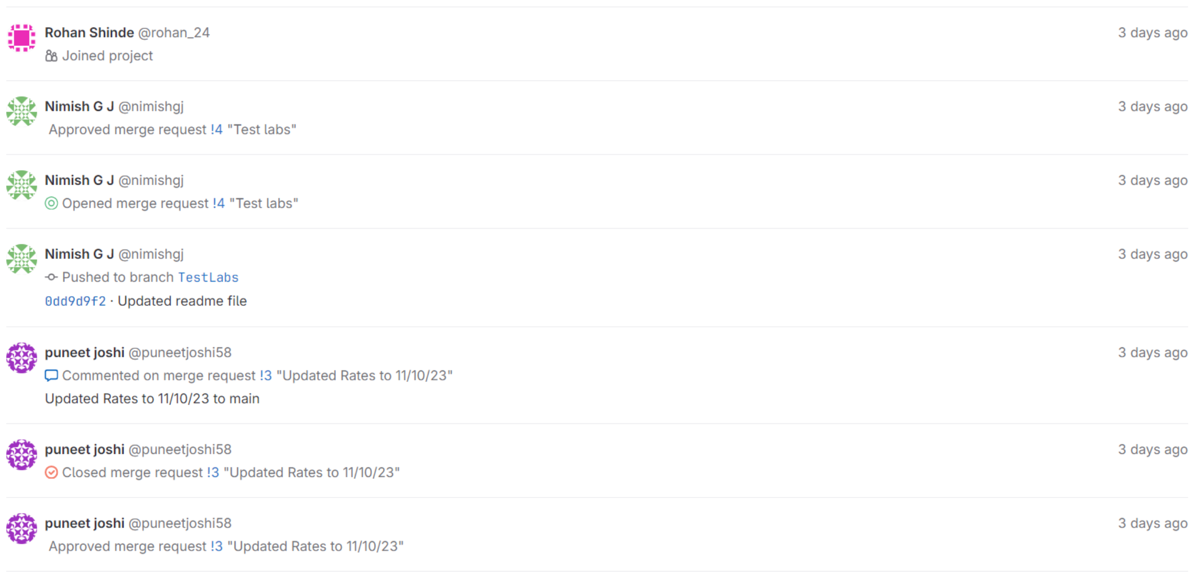
puneet joshi @puneetjoshi58 3 days ago
Closed merge request !6 "Updated currencies."

DANISH SAJAN @2gi20cs035 3 days ago
Opened merge request !6 "Updated currencies."

puneet joshi @puneetjoshi58 3 days ago
Closed merge request !5 "Update CurrencyConversion.py"

puneet joshi @puneetjoshi58 3 days ago
Deleted branch 2gi20cs035-main-patch-03603

DANISH SAJAN @2gi20cs035 3 days ago
Pushed to branch `TestLabs`
`0dae1b02` · Updated currencies.



Rohan Shinde @rohan_24 3 days ago
Joined project

Nimish G J @nimishgj 3 days ago
Approved merge request !4 "Test labs"

Nimish G J @nimishgj 3 days ago
Opened merge request !4 "Test labs"

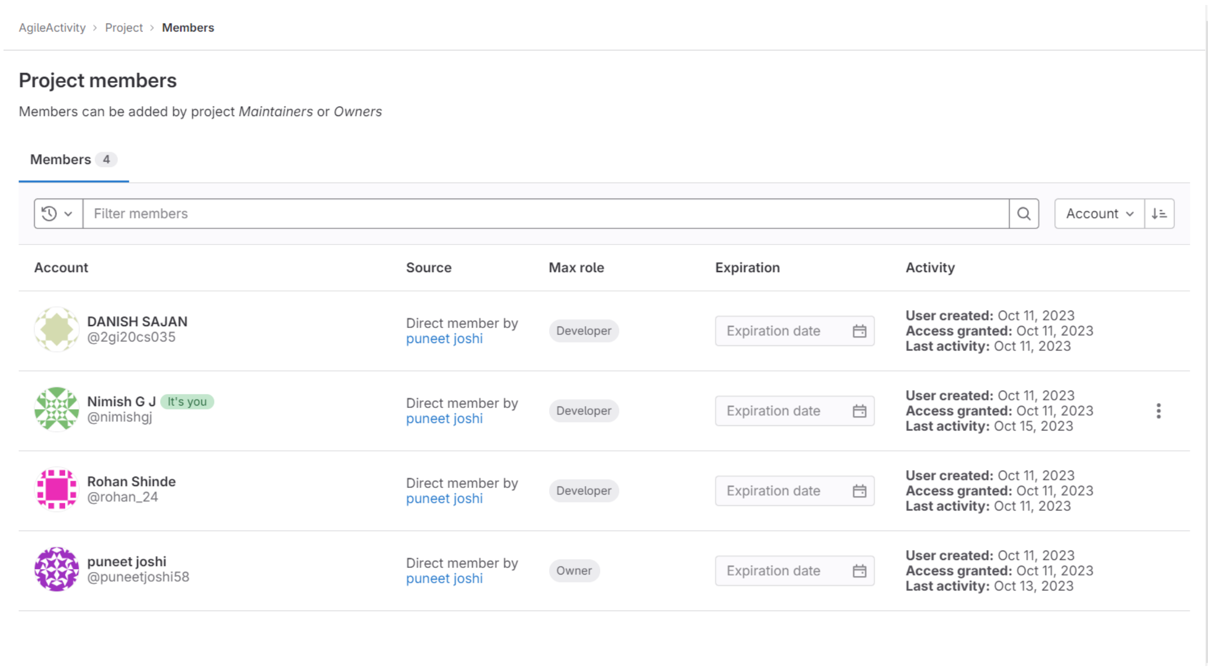
Nimish G J @nimishgj 3 days ago
Pushed to branch `TestLabs`
`0dd9d9f2` · Updated readme file

puneet joshi @puneetjoshi58 3 days ago
Commented on merge request !3 "Updated Rates to 11/10/23"
Updated Rates to 11/10/23 to main

puneet joshi @puneetjoshi58 3 days ago
Closed merge request !3 "Updated Rates to 11/10/23"

puneet joshi @puneetjoshi58 3 days ago
Approved merge request !3 "Updated Rates to 11/10/23"

Members Panel:



AgileActivity › Project › Members

Project members

Members can be added by project *Maintainers* or *Owners*

Members 4

Filter members

Account

Account	Source	Max role	Expiration	Activity
DANISH SAJAN @2gi20cs035	Direct member by puneet joshi	Developer	Expiration date	**User created:** Oct 11, 2023 **Access granted:** Oct 11, 2023 **Last activity:** Oct 11, 2023
Nimish G J It's you @nimishgj	Direct member by puneet joshi	Developer	Expiration date	**User created:** Oct 11, 2023 **Access granted:** Oct 11, 2023 **Last activity:** Oct 15, 2023
Rohan Shinde @rohan_24	Direct member by puneet joshi	Developer	Expiration date	**User created:** Oct 11, 2023 **Access granted:** Oct 11, 2023 **Last activity:** Oct 11, 2023
puneet joshi @puneetjoshi58	Direct member by puneet joshi	Owner	Expiration date	**User created:** Oct 11, 2023 **Access granted:** Oct 11, 2023 **Last activity:** Oct 13, 2023

Advantages of Using GitLab:

- Comprehensive DevOps Platform: GitLab offers an all-in-one solution that includes version control, CI/CD, issue tracking, code review, and more. This integration simplifies the development and deployment process.
- Open Source and Self-Hosted Options: GitLab is available as both an open-source community edition and a self-hosted enterprise edition. This flexibility allows organizations to choose the version that best suits their needs.
- Strong Version Control: GitLab's core functionality is built on Git, a robust and widely adopted version control system. This allows for efficient code collaboration and management.
- Built-in Continuous Integration and Deployment: GitLab provides an integrated CI/CD pipeline, making it easy to automate building, testing, and deploying code changes. This streamlines development workflows and ensures code quality
- Scalability: GitLab can accommodate both small and large development teams. It can handle a wide range of projects, from small personal repositories to enterprise-level solutions.
- Robust Security Features: GitLab includes built-in security scanning tools (SAST, DAST) that help identify and mitigate security vulnerabilities in the code. It also offers features like merge request approvals and access controls for secure collaboration.
- Flexibility: GitLab supports different types of repositories, including Git repositories, container registries, and package registries. It also allows you to integrate with various third-party tools and services.
- Issue Tracking and Project Management: GitLab offers an integrated issue tracking system and project management tools, simplifying task management and keeping everything related to a project in one place.
- Community and Support: GitLab has a large and active community, providing access to a wealth of knowledge and support. Additionally, paid enterprise versions offer professional support options.
- Customization: GitLab allows for extensive customization, enabling organizations to tailor their development environment to their specific needs.

Disadvantages of Using GitLab:

- Learning Curve: For new users, GitLab can have a steep learning curve, especially if they are new to Git and the concept of version control.
- Resource Intensive: Running a self-hosted GitLab instance can be resource-intensive, both in terms of hardware and maintenance. Smaller organizations may find this challenging.
- Complexity: While the comprehensive feature set is an advantage, it can also be a drawback for simple projects. GitLab might offer more functionality than what is needed, making it potentially overwhelming.
- Licensing Costs: While the community edition is open source and free to use, the enterprise edition comes with licensing costs. This may not be suitable for budget-constrained organizations.
- Customization Complexity: While customization is a benefit, extensive customization can lead to complexity and difficulties with version upgrades.
- Integration Challenges: Integrating GitLab with existing tools and services can sometimes be challenging, particularly when dealing with legacy systems.
- Collaboration Overhead: GitLab encourages structured collaboration, which can be a disadvantage if your development process is informal or if you prefer more ad-hoc collaboration.
- Competitive Alternatives: Depending on your specific needs, there might be alternative tools that better suit your requirements. It's essential to evaluate other options before committing to GitLab.

Conclusion:

In conclusion, using GitLab as a software development and collaboration platform offers a wide range of benefits and advantages. GitLab simplifies and streamlines many aspects of the development process, providing a comprehensive DevOps solution that encompasses version control, continuous integration, issue tracking, code review, and more. Its open-source and self-hosted options cater to organizations of various sizes, making it a flexible choice for version control and project management.

GitLab's strong integration of Git for version control, coupled with its built-in CI/CD pipelines, ensures efficient code collaboration and automates the build and deployment process, improving code quality and accelerating development workflows. The platform's robust security features, including security scanning tools, access controls, and merge request approvals, enhance code security and collaboration, which is crucial in today's software development landscape.

References:

- https://en.wikipedia.org/wiki/GitLab
- https://docs.gitlab.com/ee/user/project/wiki/
- https://www.tutorialspoint.com/gitlab/gitlab_wiki_pages.htm