THE UNIVERSITY OF DODOMA



**COLLEGE OF INFORMATICS AND VIRTUAL EDUCATION**

**CS 321: GROUP ASSIGNMENT 3**

**INSTRUCTOR: Mr. MINJA, GODBLESS G.**

**GROUP MEMBERS:**

|  |  |  |
| --- | --- | --- |
| S/N | NAME OF STUDENTS | REGISTRATION NUMBER |
| 1 | RAPHAELA MICHAEL | T/UDOM/2017/02774 |
| 2 | RENATUS SAMWEL | T/UDOM/2017/02775 |
| 3 | NICETAS SHAYO | T/UDOM/2017/02771 |
| 4 | THOMAS STEPHANO | T/UDOM/2017/02778 |
| 5 | GODFREY NICKODEM | T/UDOM/2017/02762 |
| 6 | SIGIFRIDY MUNUO | T/UDOM/2017/02777 |

LIST AND SHORTLY DESCRIBE ANY THREE TOOLS THAT ARE CURRENTLY BEING USED BY OPEN-SOURCE DEVELOPMENT TEAMS FOR COMMUNICATION PURPOSES.

* [**MATTERMOST**](https://www.goodfirms.co/software/mattermost)

Mattermost is an enterprise-level free and open-source team communication software that is flexible and enables teams to collaborate with high-end security. It is a highly scalable software that can be customized to your business needs as it provides full access to the source code. It allows deploying it on-premise and cloud both and works seamlessly on multiple platforms and devices. It is written in Golang and React and runs as a single Linux binary with database systems like MySQL or PostgreSQL

**Key Features:**

* Allows building best workflows for your business organization
* Provides complete ownership and control over your company IP and customers’ data
* Safeguards against unexpected security hacks and threats
* Allows you to search messaging history easily
* Ability to share voice, screen, files, images, links, and videos
* Supports multiple languages
* Allows you to customize your desktop, emails, and mobile notifications
* Can integrate with external plugins, tools, and applications
* Allows performance and system monitoring
* [**ZULIP**](https://www.goodfirms.co/software/zulip)

Zulip is a 100% free and open-source team communication software that is built by a community of developers located all around the world. It includes 120,000 words of developer documentation, a high-quality, customizable code base, and a highly active community, making it easy for the developers to tweak or extend the software as per the business needs. It facilitates threading messages based on topics, and that makes large group discussions extremely manageable.

**Key Features**

* Includes extremely UI friendly real-time chatting features
* Can integrate with desktop and mobile applications, and APIs
* Provides visibility over availability of other team members
* Allows threading conversations and accessing when required
* Includes video calling facility
* Holds capability of exporting data and converting hosted solution from on-premise anytime
* Open communities available for help and support
* Includes reporting features allowing to analyze the communication patterns of the team members
* [**WIRE**](https://www.goodfirms.co/software/wire)

Wire is an open-source team communication software that is free to use for 30 days. This tool communication tool is well known for its high-end secure environment, as it is protected with end-to-end encryption. It supports multiple devices and platforms, and so the users can stay connected wherever they are. It has a unique guest room feature that allows you to communicate with the users in a secure manner without revealing credentials.

**Key Features**

* Allows audio/video calls and sharing files
* Can define user & group permissions
* Works on iOS, Android, Windows operating systems
* Facilitates read receipts, and status updates
* Includes conversation folders
* High-end security features like fingerprint access, device access management, and contact verification

LIST AND SHORTLY DESCRIBE ANY THREE LATEST VERSION CONTROL SYSTEMS THAT ARE CURRENTLY USED FOR OPEN-SOURCE DEVELOPMENT.

* **GIT**

Git is considered to be a newer, and faster emerging star when it comes to version control systems. First developed by the creator of Linux kernel, Linus Torvalds, Git has begun to take the community for web development and system administration by storm, offering a largely different form of control. Here, there is no singular centralized code base that the code can be pulled from, and different branches are responsible for hosting different areas of the code. As a fast and efficient system, many system administrators and open-source projects use Git to power their repositories

* **MERCURIAL**

This is another form of version control system, similar to Git. It was designed initially as a source for larger development programs, often outside of the scope of most system admins, independent web developers and designers. Mercurial is a very fast and efficient application. The creators designed the software with performance as the core feature.  
Mercurial is known for its efficiency in handling projects of all sizes. It is a free and distributed control management service that provides a simple and intuitive user interface. Developers and enterprises adore Mercurial for its backup system, search functionality, project tracking and management, data import and export, and data migration tool. It also features workflow management, history tracking, security management, access controls and more

* **BAZAAR**

Bazaar is distributed version control system, which also provides a great, friendly user experience. Bazaar is unique that it can be deployed either with a central code base or as a distributed code base. It is the most versatile version control system that supports various different forms of workflow, from centralized to decentralized, and with a number of different variations acknowledged throughout. One of the greatest features of Bazaar is that you can access a very detailed level of control in its setup. Bazaar can be used to fit in with almost any scenario and this is incredibly useful for most projects and admins because it is so easy to adapt and deal with. It can also be easily embedded into projects that already exist.

LIST AND SHORTLY DESCRIBE THE PROS AND CONS OF EACH OF THE FOLLOWING TYPES OF VERSION CONTROL SYSTEMS

* **CENTRALIZED VERSION CONTROL SYSTEMS**

Centralized Version Control Systems were developed to record changes in a central system and enable developers to collaborate on other systems.

**Advantages of Centralized Version Control Systems:**

* Relatively easy to set up
* Provides transparency
* Enable admins control the workflow

**Disadvantages of Centralized Version Control Systems:**

* If the main server goes down, developers can’t save versioned changes
* Remote commits are slow
* Unsolicited changes might ruin development
* If the central database is corrupted, the entire history could be lost
* **DISTRIBUTED VERSION CONTROL SYSTEMS**

Distributed Version Control Systems (DVCSs) don’t rely on a central server. They allow developers to clone the repository and work on that version. Develops will have the entire history of the project on their own hard drives

**Advantages of Distributed Version Control Systems**:

* Because of local commits, the full history is always available
* No need to access a remote server (faster access)
* Ability to push your changes continuously
* Saves time, especially with SSH keys
* Good for projects with off-shore developers

**Disadvantage of** **Distributed Version Control Systems:**

* It may not always be obvious who did the most recent change
* File locking doesn’t allow different developers to work on the same piece of code simultaneously. It helps to avoid merge conflicts, but slows down development
* DVCS enables you to clone the repository
* Managing non-mergeable files is contrary to the DVCS concept
* Working with a lot of binary files requires a huge amount of space.

CIRCUMSTANCES OF USING CVCS AND DVCS

Centralized Version Control is the simplest system with the concept of 1 central repository which servers provides latest code to the all the clients across the globe

Distributed Version Control provides flexibility and has emerged with the concept that everyone has their own repository, they don’t just check out the snapshot of the code they fully mirror the central repository.

* CVCS is easy to understand whereas DVCS has some complex process for beginners.
* CVCS is dependent on the access to the server whereas DVCS provides the benefits to work offline. Everything except push and pull the code can be done without an internet connection.
* CVCS is easy to administrate and has more control over users and access as it is server from one place.
* DVCS is comparatively fast comparing to CVCS as you don’t have to contact the central server for every command. DVCS just takes much time on the first check-out as its mirroring the central repository on your local.
* If your project has a very long history and change-sets then downloading the entire history can take an unreasonable amount of time and disk space in DVCS whereas CVCS allows you to checkout only few lines of code if you just need to work on few modules.
* DVCS provides a powerful and detailed change tracking, which means fewer conflicts at the time of merge.
* DVCS gives an ability that developers can share changes with one or two other members of team at a time if they want to get some feedback before showing the changes to everyone.
* The revisions in DVCS are typical big guides (like fa333b7rer96cd6d3b0037d660) – it’s not incremental numbers (which is provided by CVCS) which make them harder to reference and remember.
* DVCS provides an advantage wherein if the main server’s repository crashes, you still have a local repository in every developer’s local space from which you can create the main repository.

**LAB WORK**

Most prominent version control system

Centralized

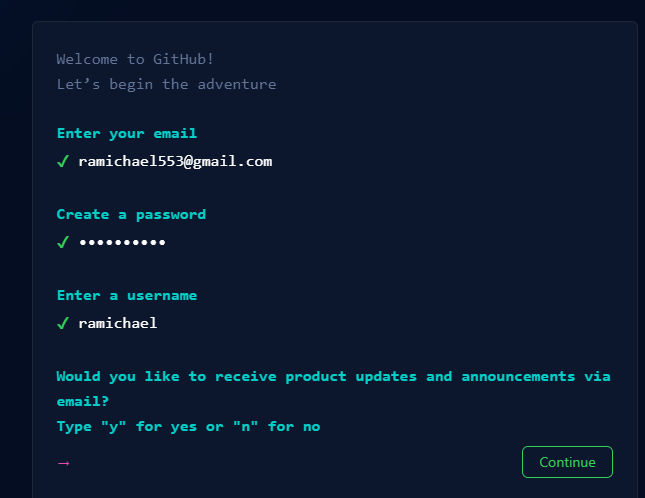
* Svn

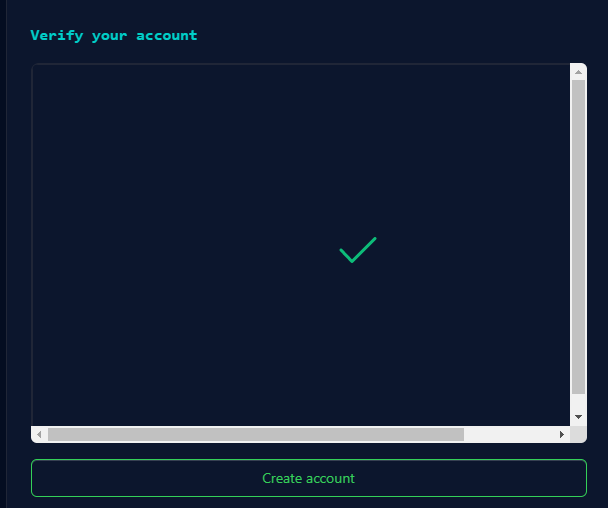
Decentralized

* Git

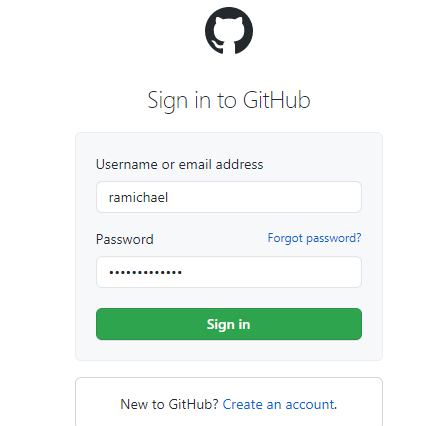
BY USING GIT

1. REGISTRATION

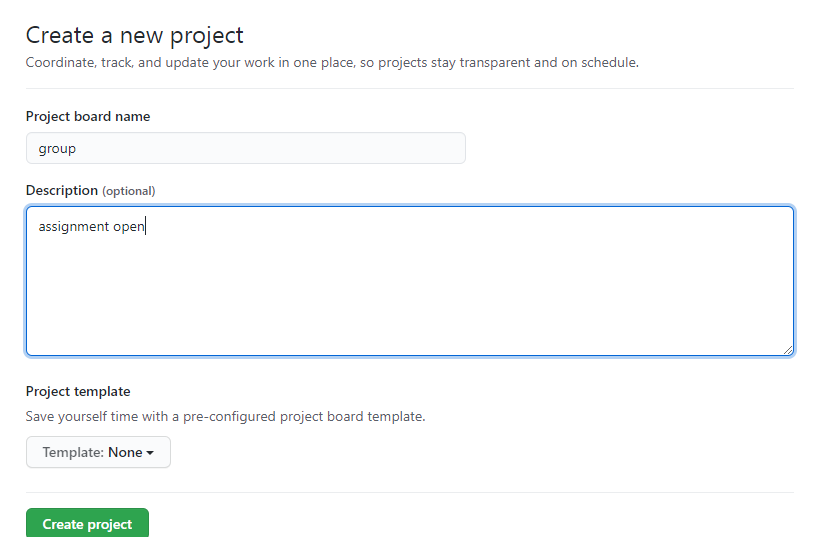


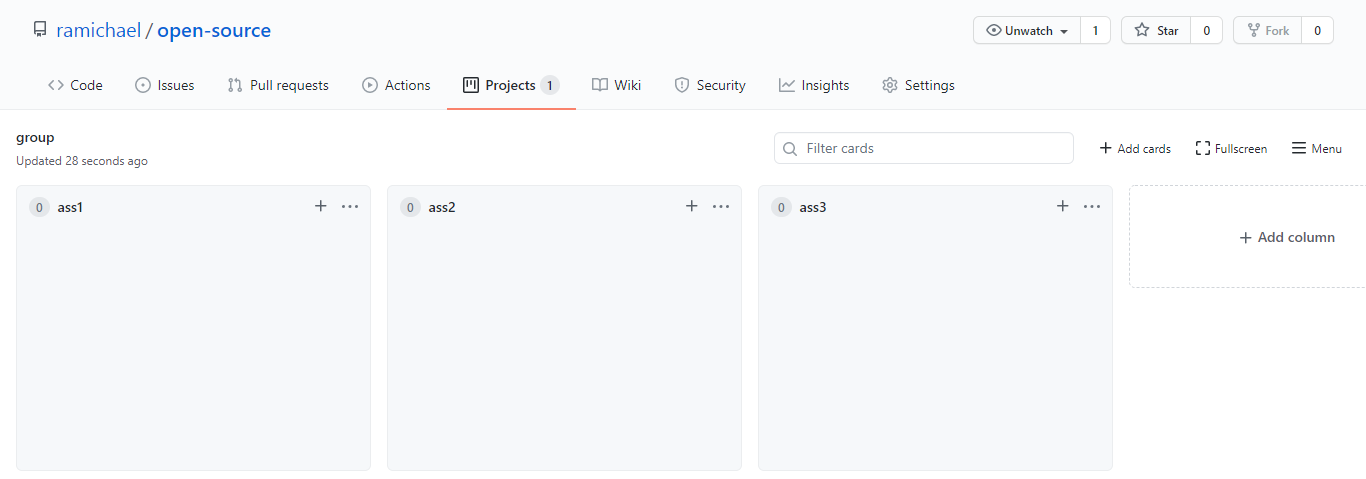


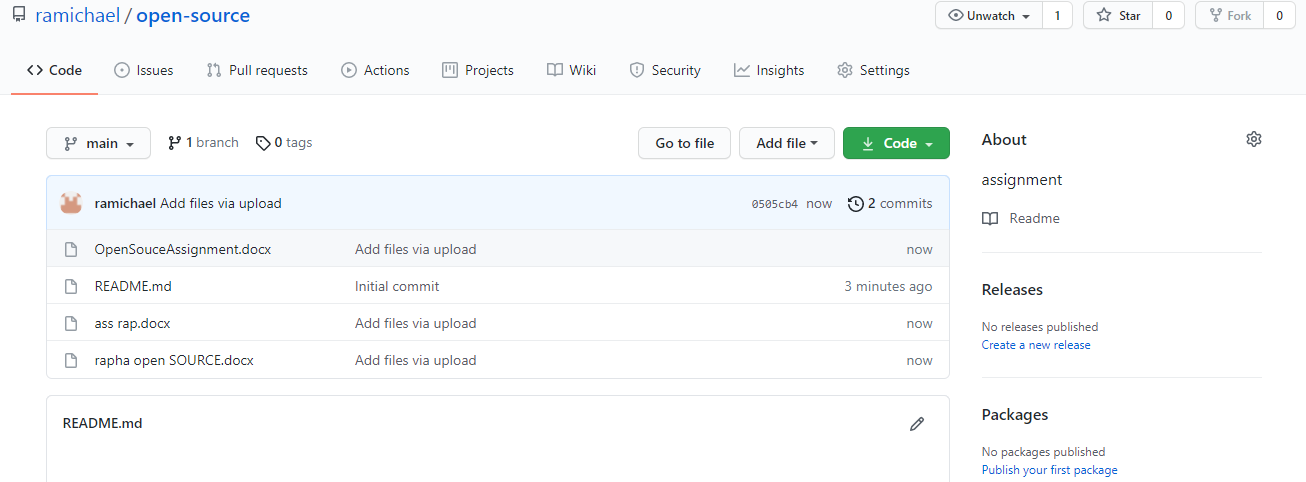
1. LOGIN



1. PROJECT CREATION







1. INVITATION TO PARTICIPANTS