Secure File Synchronization Tool with Version Management

Foundation of computer programming

Team members:

- Omar Alfarouq BOUHADI
- Zakaria CHOUKRI
- Ayman YOUSS

Introduction:

This project introduces a local version control system resembling Git, developed in C++ with a Qt-based graphical user interface. The system facilitates version tracking and commit management in a local development environment. Key functionalities, including init, add, commit, pull, and revert, are implemented to offer users an intuitive and efficient version control solution. The report details the project's specific objectives, the technology stack, system architecture, features, and user benefits. The successful implementation of a functional system with a user-friendly interface addresses the need for a straightforward version control solution in local development scenarios.

Technologies used:

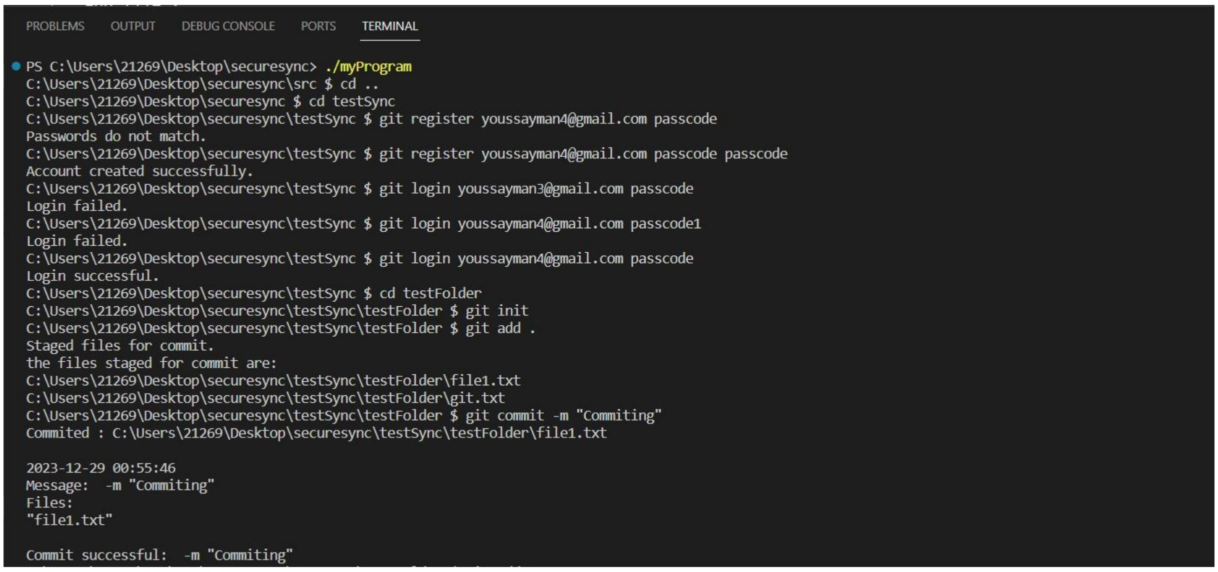
In the development of the project, we primarily utilized C++ for the core functionality and Qt for the graphical user interface (GUI) aspects. C++ provided the necessary power and flexibility for building a robust local version control system, while Qt simplified the creation of an intuitive and user-friendly interface, enhancing the overall user experience.

Getting started:

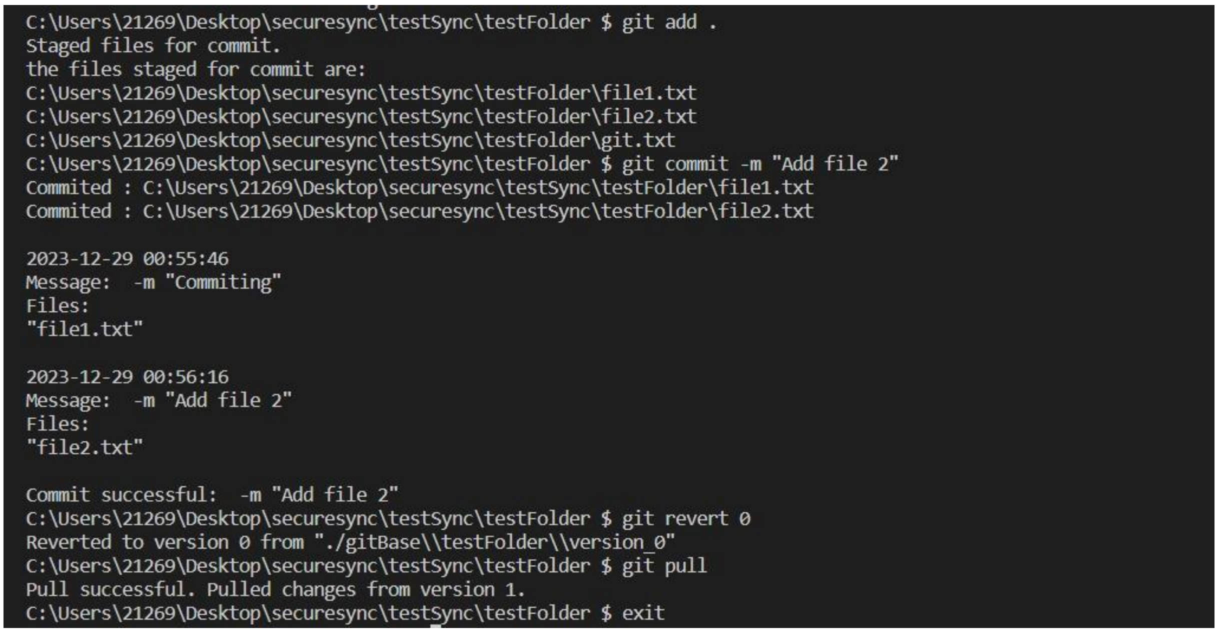
1. Execute the provided shell script: compile.sh to build the project. This script creates the myprogram.exe executable, a key component of our system.
2. After successful compilation, run the myprogram.exe executable. This step initializes the program and sets the stage for command input.
3. The program opens a command prompt, providing an interactive environment for users to input commands. Familiarize yourself with the supported commands to navigate and utilize the features of our local version control system.
4. Enter supported commands in the command prompt to perform various actions, such as git add, commit, or pull.

Command line interface:

- Navigating to the desired directory using the cd command.
- Register and log in using the user’s credentials.
- Using git init to initialize the repository.
- Using git add to add desired files to the stage for a commit later.
- Using git commit with a message to commit the files in the stage.
- Using git revert 0 to come back to version 0 of the repository.
- Using git pull to get the up-to-date version of the repository.
- Using exit to close the command line interface.



```
PS C:\Users\21269\Desktop\securesync> ./myProgram
C:\Users\21269\Desktop\securesync\src $ cd ..
C:\Users\21269\Desktop\securesync $ cd testSync
C:\Users\21269\Desktop\securesync\testSync $ git register youssayman4@gmail.com passcode
Passwords do not match.
C:\Users\21269\Desktop\securesync\testSync $ git register youssayman4@gmail.com passcode passcode
Account created successfully.
C:\Users\21269\Desktop\securesync\testSync $ git login youssayman3@gmail.com passcode
Login failed.
C:\Users\21269\Desktop\securesync\testSync $ git login youssayman4@gmail.com passcode1
Login failed.
C:\Users\21269\Desktop\securesync\testSync $ git login youssayman4@gmail.com passcode
Login successful.
C:\Users\21269\Desktop\securesync\testSync $ cd testFolder
C:\Users\21269\Desktop\securesync\testSync\testFolder $ git init
C:\Users\21269\Desktop\securesync\testSync\testFolder $ git add .
Staged files for commit.
the files staged for commit are:
C:\Users\21269\Desktop\securesync\testSync\testFolder\file1.txt
C:\Users\21269\Desktop\securesync\testSync\testFolder\git.txt
C:\Users\21269\Desktop\securesync\testSync\testFolder $ git commit -m "Commiting"
Commited : C:\Users\21269\Desktop\securesync\testSync\testFolder\file1.txt

2023-12-29 00:55:46
Message:  -m "Commiting"
Files:
"file1.txt"

Commit successful:  -m "Commiting"
```

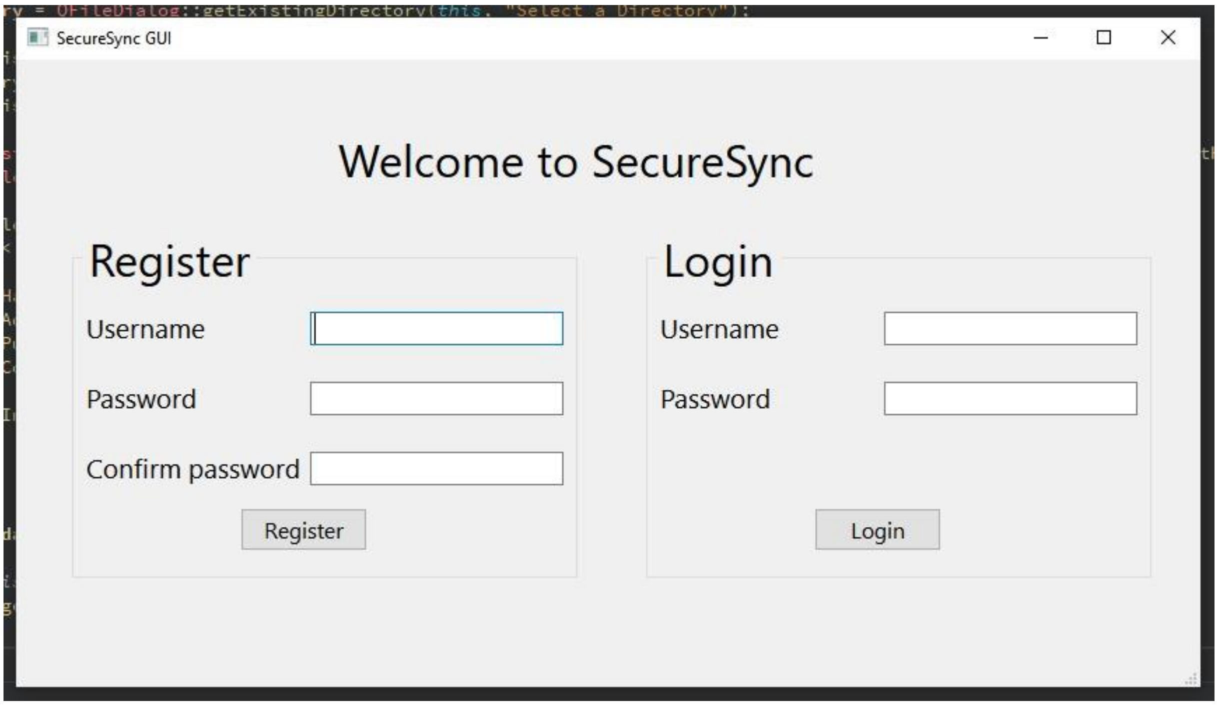


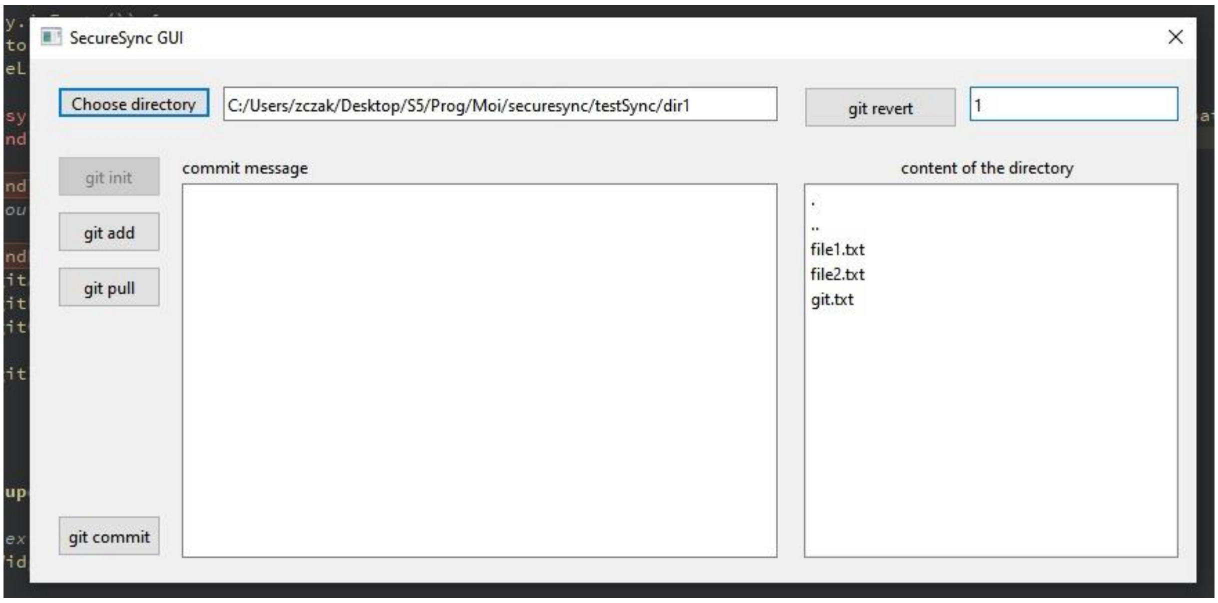
```
C:\Users\21269\Desktop\securesync\testSync\testFolder $ git add .
Staged files for commit.
the files staged for commit are:
C:\Users\21269\Desktop\securesync\testSync\testFolder\file1.txt
C:\Users\21269\Desktop\securesync\testSync\testFolder\file2.txt
C:\Users\21269\Desktop\securesync\testSync\testFolder\git.txt
C:\Users\21269\Desktop\securesync\testSync\testFolder $ git commit -m "Add file 2"
Commited : C:\Users\21269\Desktop\securesync\testSync\testFolder\file1.txt
Commited : C:\Users\21269\Desktop\securesync\testSync\testFolder\file2.txt

2023-12-29 00:55:46
Message:  -m "Commiting"
Files:
"file1.txt"

2023-12-29 00:56:16
Message:  -m "Add file 2"
Files:
"file2.txt"

Commit successful:  -m "Add file 2"
C:\Users\21269\Desktop\securesync\testSync\testFolder $ git revert 0
Reverted to version 0 from "./gitBase\\testFolder\\version_0"
C:\Users\21269\Desktop\securesync\testSync\testFolder $ git pull
Pull successful. Pulled changes from version 1.
C:\Users\21269\Desktop\securesync\testSync\testFolder $ exit
```

<u>**Graphical User Interface:**</u>

- User Registration:
  - New users can create accounts by providing a valid email and password.
- Login Feature:
  - Existing users can log in using their registered email and password.
- Git Operations :
  - The user interface presents buttons for essential Git commands.
    - Init: Set up a new Git repository.
    - Add: Stage files for commit.
    - Commit: Save staged changes with a descriptive message.
    - Pull: Retrieve changes from a remote repository.
    - Revert**:** Roll back to a previous version, providing version number.
- Intuitive Navigation :
  - Simplified buttons ensure a user-friendly experience for version control tasks.

- Users can easily navigate and select directories using a dedicated button, enhancing flexibility in managing repositories.


SecureSync GUI
Welcome to SecureSync
Register
Username
Password
Confirm password
Register
Login
Username
Password
Login


SecureSync GUI
Choose directory
C:/Users/zczak/Desktop/S5/Prog/Moi/securesync/testSync/dir1
git revert
1
git init
git add
git pull
git commit
commit message
content of the directory
.
..
file1.txt
file2.txt
git.txt