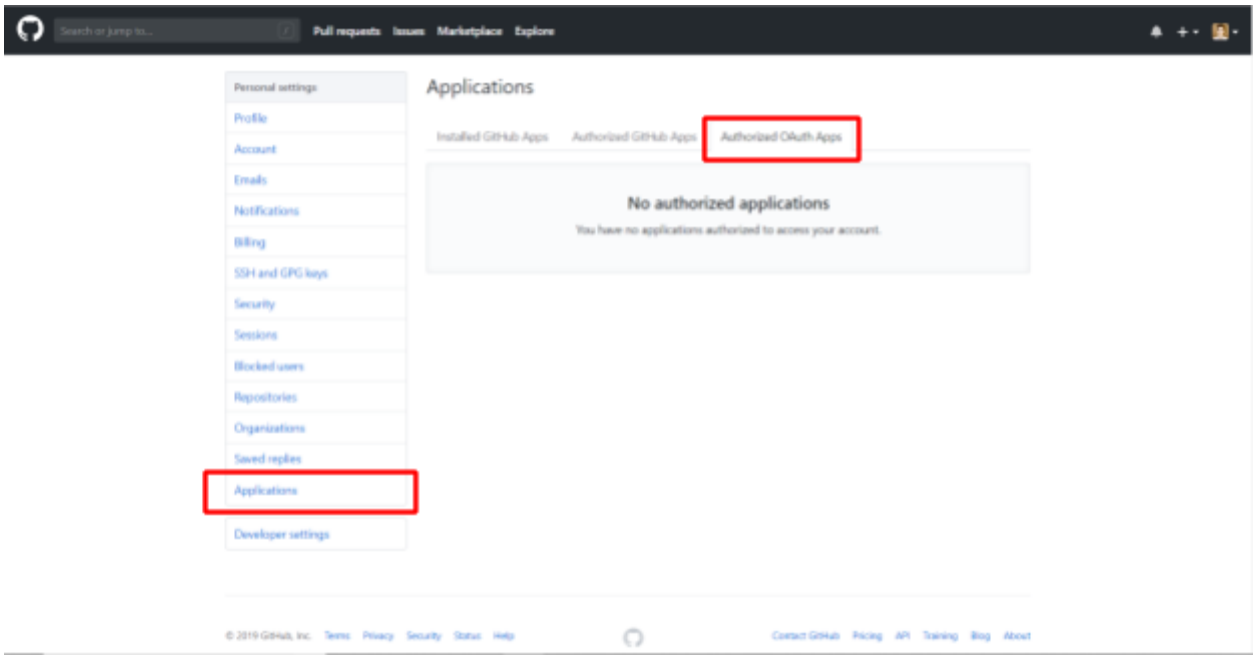
Getting Started:

To start GitUp, first navigate to the following GitHub Repository URL: [“GitUp download URL”]
Here you can download GitUp, which will provide you with an executable to launch GitUp.

Login:

To login to GitUp please provide a GitHub username and password on your first login. GitUp uses GitHub to automatically create and host repositories for your projects and we do not store your login information! (See our open source code[link]). On your first login you we will create an authorization token for your account to GitUp, you can manage GitUp’s permissions at any time from your account:



That’s it! GitUp is ready to use, allowing you to focus on creating amazing projects while we handle all the details.

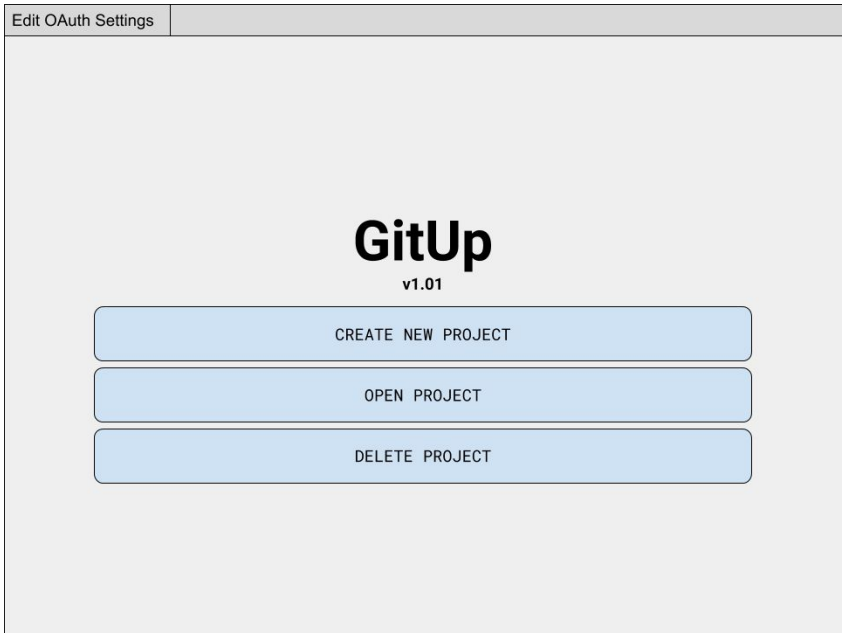
(Of course if you want to edit GitUp’s permissions or change your GitHub account, simply click “Edit OAuth Settings” in the top-right corner of the GitUp home screen.)

Data Storage

Gitup stores a variety of types of data such as the names of the projects the user has created, which ones are active on the current machine, the paths to the root folder of

each project, and the URL to the github repository. This data is stored in a file called projects.csv in directory gitup is installed to.

The Main Screen



Creating a Project

After setting up OAuth and on all subsequent times the application is started the user is presented this screen.

Clicking the “Create new project” button launches a dialog that asks the user to input a name for the project. Upon entering a name and clicking “Create” a new git repository is created on the user’s machine and pushed to Github.

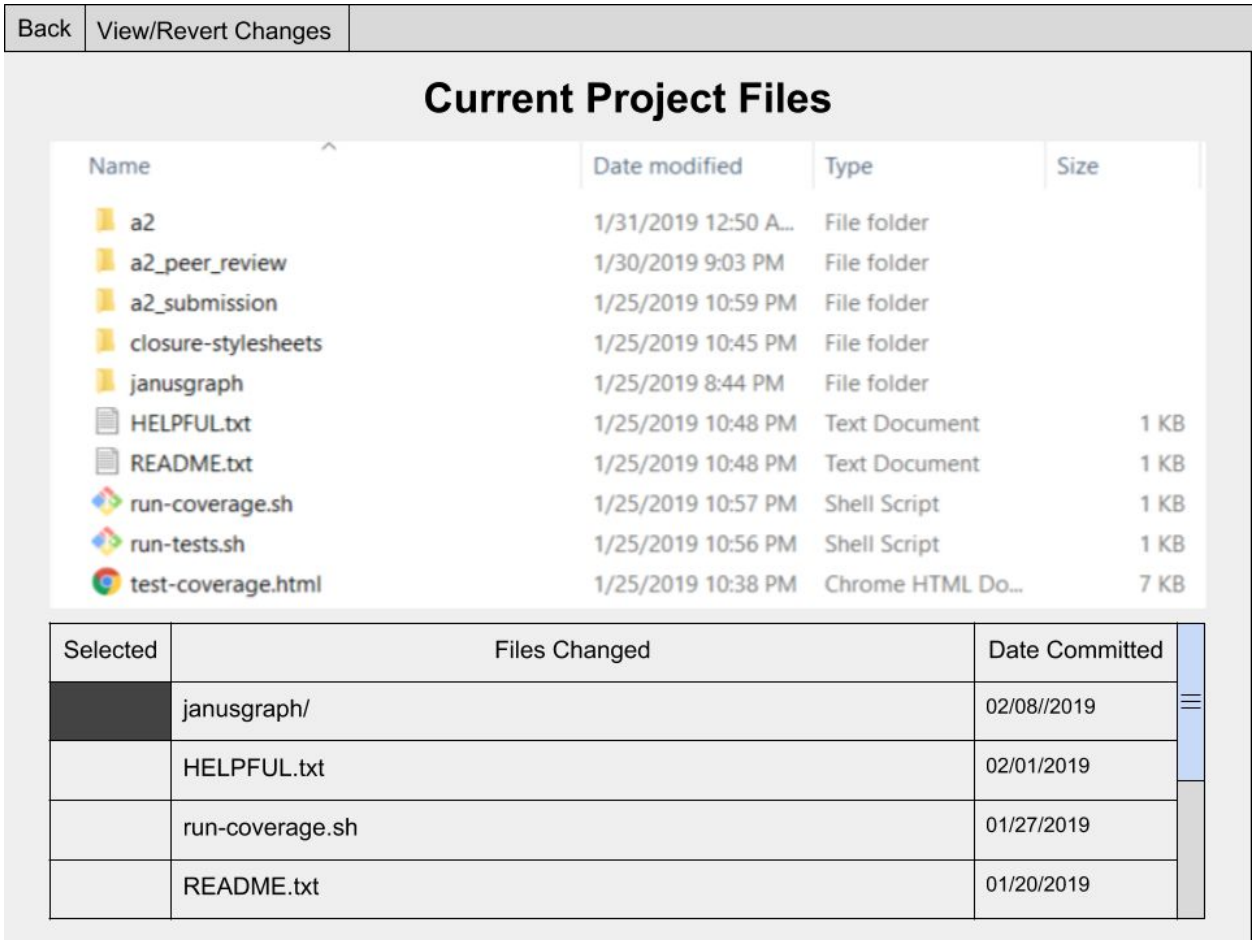
Deleting a project

Deleting a project is as simple as clicking the “Delete project” button and then selecting the project to delete. When this action is taken a dialog box pops up asking the user if they want to delete the project on Github as well, and their choice is honored. Deleting the local files for a project will not delete the project on Github but will tell the application to stop searching for changes to these projects on this machine.

Opening a project

Clicking the “Open project” button and selecting a project takes the user to the project screen for the project the user selected.

The Project Screen



The View/Revert Changes Screen

Selecting a file and clicking the “View/Revert Changes” button from the project screen allows the user to see a file’s history. The diff between the two commits is displayed and the user sees insertions in green and deletions in red.

Back | Change Commits | Revert to Pre

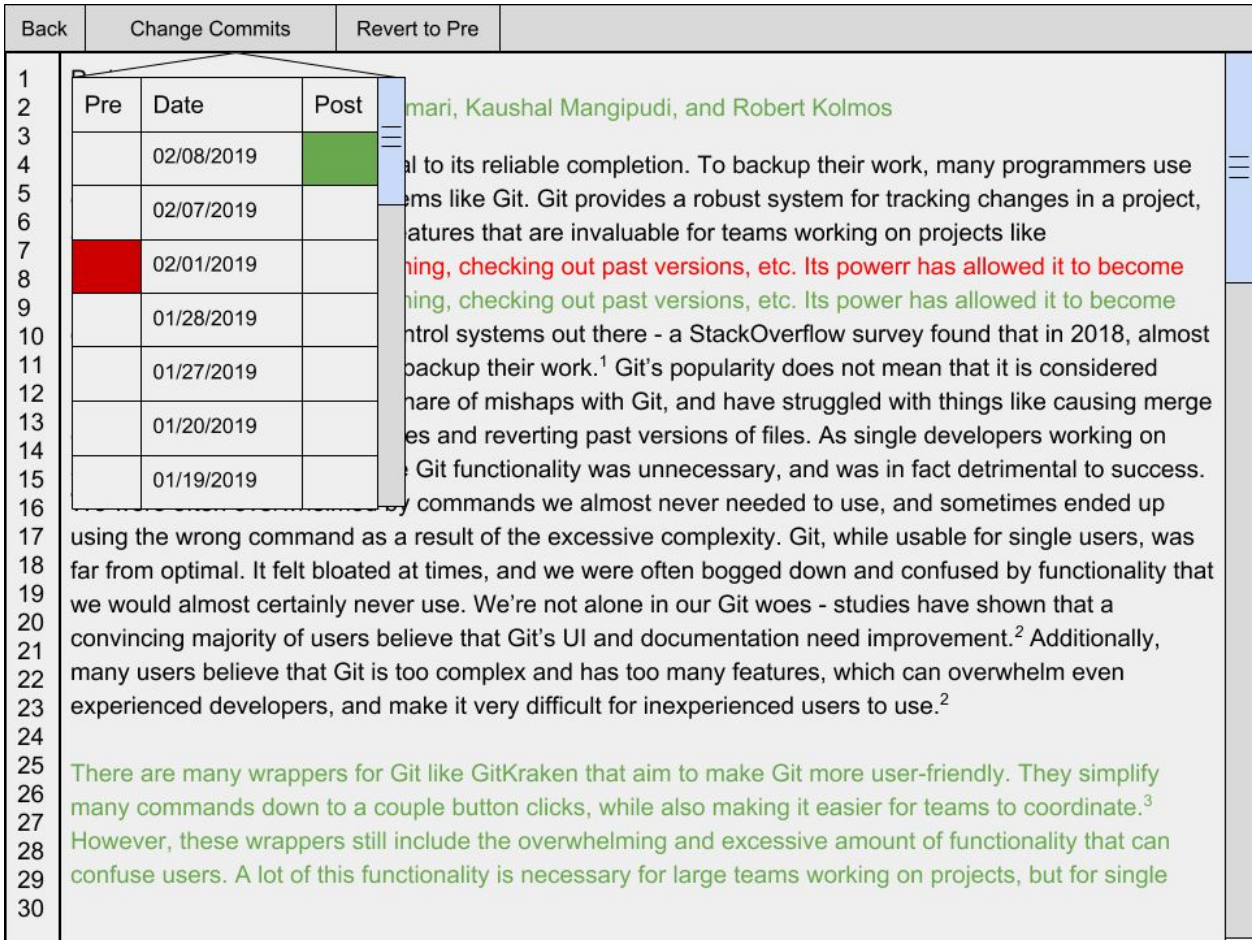
Project Proposal
By: Kamden Chew, Gerard Gaimari, Kaushal Mangipudi, and Robert Kolmos

Backing up a project is essential to its reliable completion. To backup their work, many programmers use distributed version control systems like Git. Git provides a robust system for tracking changes in a project, and comes with a plethora of features that are invaluable for teams working on projects like pushing/pulling, tagging, branching, checking out past versions, etc. Its powerr has allowed it to become
pushing/pulling, tagging, branching, checking out past versions, etc. Its power has allowed it to become one of the dominant version control systems out there - a StackOverflow survey found that in 2018, almost 90% of developers used Git to backup their work.[1] Git's popularity does not mean that it is considered perfect. We have had our fair share of mishaps with Git, and have struggled with things like causing merge conflicts by failing to pull changes and reverting past versions of files. As single developers working on small scale projects, a lot of the Git functionality was unnecessary, and was in fact detrimental to success. We were often overwhelmed by commands we almost never needed to use, and sometimes ended up using the wrong command as a result of the excessive complexity. Git, while usable for single users, was far from optimal. It felt bloated at times, and we were often bogged down and confused by functionality that we would almost certainly never use. We're not alone in our Git woes - studies have shown that a convincing majority of users believe that Git's UI and documentation need improvement.[2] Additionally, many users believe that Git is too complex and has too many features, which can overwhelm even experienced developers, and make it very difficult for inexperienced users to use.[2]

There are many wrappers for Git like GitKraken that aim to make Git more user-friendly. They simplify many commands down to a couple button clicks, while also making it easier for teams to coordinate.[3] However, these wrappers still include the overwhelming and excessive amount of functionality that can confuse users. A lot of this functionality is necessary for large teams working on projects, but for single

By clicking "Change Commits" the user sees a menu allowing them to select a pre and post commit. Clicking "Revert to Pre" will allow the user to set the current file to

whatever previous commit they currently have selected in the “Change Commits” table.



Syncing changes

You don’t need to worry about syncing changes! GitUp will automatically back up your work whenever you change. If syncing fails, we will give you a warning about which files could potentially become out of sync with the backed up version.

Important Notes

Since the projects are on your GitHub account, you can modify your project using git. However, we strongly discourage doing this, as GitUp will handle all aspects of backing up your work and vewing/reverting past versions for you. Also, proceed with caution whenever you get a warning that something could potentially become out of sync! If a file becomes out of sync, GitUp will automatically attempt to resolve it, though you may lose work when it does so.

References

[1]https://docs.python.org/3/library/tk.html
[2]https://gitpython.readthedocs.io/en/stable/
[3]http://man7.org/linux/man-pages/man7/inotify.7.html