GitUp

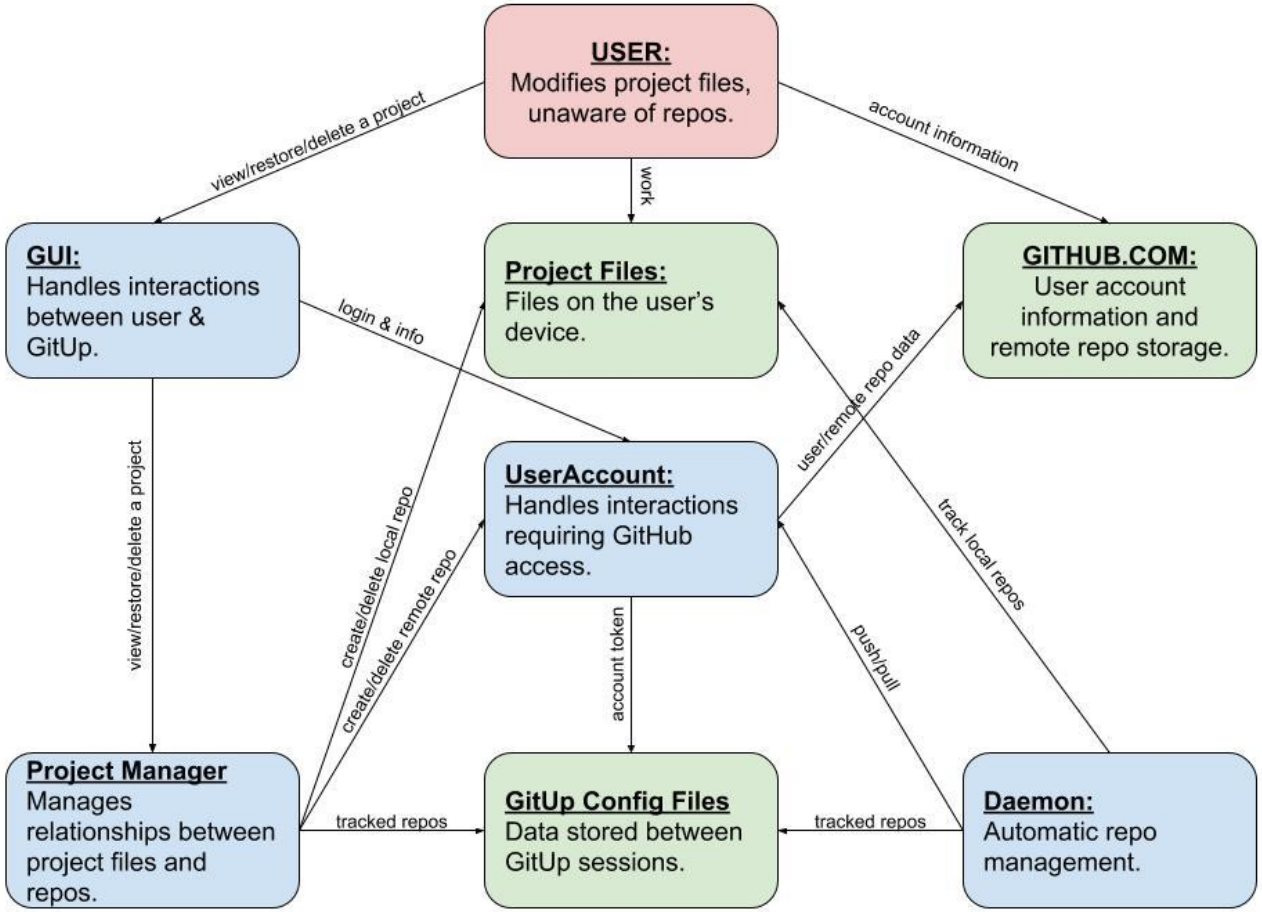
Kaushal Mangipudi, Gerard Gaimari, Kamden Chew, Robert Kolmos

Why GitUp?

- Non-developers working on software projects need a backup tool that:
  - Robustly tracks past versions of files for a long time
  - Allows comparing and reverting to past versions of files
  - Is easy to use and out of the way
- Current backup tools don’t satisfy all these needs
  - Cloud sync services don’t offer long term version tracking
  - Cloud backup services don’t allow you to compare past versions with one another
  - Version Control Systems and their derivatives are too complex for non-developers
- GitUp is the only tool that will meet all three of these needs!

Module Architecture

with key interactions



Demo

The Daemon

- Is started whenever the system reboots.
  - Want it to always be running, but the user can manually kill it
- Responsible maintaining local repositories, detecting changes, commiting files, and pushing/pulling.
- Uses Linux’s inotify API to detect file changes.

Automatic Committing

- Commits the changes to a file every time an event that modifies that file occurs.
- Works well when the user's editor doesn't autosave (Vim, Emacs, etc.)
- Creates a large number of commits when the editor does autosave (Eclipse, Intellij, etc.)
- Stretch Goal: Squash commits with very similar timestamps (i.e. < 1 min)

Automatic Pushing/Pulling

- Difficult to minimize merge conflicts without user action.
  - Lots of actions lead to merge conflicts.
  - Solutions are heuristics, not perfect.
- Currently pushing/pulling is time based (5 mins, 30 sec in demo)
- Other ideas:
  - Pulling: Monitor file opens, in conjunction with time since last pull
  - Pushing: Monitor file are closes.

Evaluation Challenges

- Initially, testers would perform a series of tasks using GitUp, Git and GitKraken
  - Compare completion times, and collect answers to comparative questions
  - Systems presented in different orders - eliminates learner bias
  - All 3 are git-based - eliminates any system discrepancies
- The design was fairly comprehensive, but...
  - Too time consuming (90+ minutes per tester)
  - Requires too large a sample size
- We can use it if we ever have time to comprehensively test GitUp in the future

Two-Tiered Evaluation Approach

- Ease of Use Testing
  - Give non-developers 30 minutes to attempt to:
    - Setup GitUp on a machine
    - Add an existing local project to GitUp
    - Add a remote GitUp project to their local machine
    - View general change history of their project and view and revert changes to specific files
    - Stop backing up a local project
  - Record time taken for each task and general feedback about usability of GitUp
- Comparison Testing
  - Get feedback from programmers familiar with version control systems
    - How does GitUp stack up against its competitors?
    - Does it have all the functionality you'd need for backing up a software project?
    - Would it have helped you back when you were inexperienced?

Takeaways

- GitUp will be the only tool available that meets all the needs of non-developers working on software projects
- It’s easy to use and out of the way design allows non-developers to focus on their code, not on backing it up
- GitUp’s two-tiered evaluation approach ensures that it will be...
  - Intuitive enough for non-developers to use
  - Functional enough to support all their needs