## Requirements

**1. Web app**

**2. Allow selection of any GitHub repository**

**3. Visualize the complete history of the file**

**For each commit**

**4. Contents of the file**

**Metadata for the change, such as:**

**5. Author**

**6. Date**

**7. commit message**

**8 . Size of the file**

**Filter results by:**

**9. File**

**10. Author**

**11. Time frame**

**12. Branch**

**13. File type (language) ?**

**14. Allow exporting the history of a file to a downloadable file: PNG, CSV, HTML**

**15. Allow login for viewing private repos**

**16. Switch between graphing by commit count and by commit volume**

## 

**Use Case 1 (UC1): Opening the webapp and searching for a repository**

* **Stakeholders and Interests**

Anybody interested in viewing the history of files in a git repo.

* **Preconditions**  
  Internet is on, browser supports js programs, github isn’t down...
* **Success Guarantee**  
  User sees a page with tabs to filter. That page contains a stream graph, files, authors, branches, etc.
* **Main Success Scenario**  
  1. User goes to the webapp.   
  2. Searches for a github repo  
  3. Is shown a stream graph and a list of “things” - can be ‘toggled’ by 3 tabs, 1) list of all files and folders in that repo, 2) list of all authors, 3) list of all branches  
  4. Can click on a folder to view the files/folders inside of that  
  5. Can click on any file to see it in detail
* **Extensions**  
  2a) User searches for a private repository and only sees it if logged in with appropriate access  
  3a) Filter based on files/folder  
   3a1) Shows list of all the files/folders and their sizes  
   3a2) Can click on one to view commits affecting that file  
   3a3) Can click on the commit to view details for that commit  
  3b) Filter based on authors  
  3c) Filter based on branches  
  4a) User clicks on an empty folder and is shown a screen saying there are no files
* **Special Requirements**  
  Boxes containing information scroll properly when the list is longer than the box  
  Color - all files, folders, etc. etc. are colored for ease of use  
  ^Color is optimized for RG colorblindness
* **Technology and Data Variation List**  
  Browser has to support our javascript libraries  
  Webapp has to support user login for extension 2a)  
  Can filter anything based on valid regex’s...
* **Frequency of Occurrence**  
  Every time they open the program to use it. Once it’s already open, they just have to search for the repo/files (and not open the webapp)

**Use Case 2 (UC2): Selecting a file in a repo to visualize the history**

* **Stakeholders and Interests**

Anybody interested in viewing the history of files in a git repo.

* **Preconditions**User has properly opened webapp and searched for the given repository. Sees a list of all files in the repo with stream graph/etc.
* **Success Guarantee**Displays all the commits of a given file, which when clicked display all the text information about the file history. Includes username, branches, project name, etc.
* **Main Success Scenario**1. User has list of all files in the repository  
  2. User clicks on the file they’re interested in  
  3. A “box” in the page is loaded that displays the commit history of the file with user names, dates, etc.   
  4. Can click on a given commit to get metadata about the file
* **Extensions**1a) Page regularly updates to reflect changes in the repo  
  2a)User can click on a file/folder that’s been deleted since they searched for the repository  
   2a,1) Brings user to an error page saying file/folder is deleted
* **Special Requirements**To load the graphs the user must be on a browser that interacts with our javascript libraries.
* **Technology and Data Variation List**Ability to scroll in boxes if they’re larger than the display allows
* **Frequency of Occurrence**Every time they use the program and are interested in viewing details for a given file/commit/branch

**Use Case 3 (UC3): Getting the history of a file/folder/repo and exporting it**

* **Stakeholders and Interests**  
  People that are interested in using this data offline (for a presentation, perhaps). Users could be researchers interested in github history of a file, managers tracking their employees’ work, etc.
* **Preconditions**  
  Has properly found the file or filtered by whatever and viewed the visualizations they want to download
* **Success Guarantee**  
  User gets a full downloaded copy of the information
* **Main Success Scenario**1) User sees graphs of information they want for a given file  
  2) Clicks download on the information/graph/etc. and gets a file containing all the info they want  
  3) Can open that file offline
* **Extensions**  
  1a) Can also download information for more than one file  
   1a1) Can download graphs based on any filters they can apply from the main UI
* **Special Requirements**  
  None
* **Technology and Data Variation List**  
  Uses the knockout.js and d3.js packages  
  Can export as image (for just a file), as CSV (just data), or as HTML (includes everything)
* **Frequency of Occurrence**  
  Rarely? Only when people care about stuff offline, like for a presentation