Assignment 1-Fully-Dressed Use Cases

Fully-Dressed Use Cases

Authors: Somn Kafley, John McDouall, Jordan Kuhn, and Steffen Moseley

Fully Dressed Use Cases - Assignment 1 (attempt #2 - this time with 16 use cases)

Use Case 1: User searches for a public repository (top right)

- **Stakeholders and Interests**
  Any user wanting to search public repository.
- **Preconditions**
  Internet is on, browser supports js programs, github isn't down...
- **Success Guarantee**
  User sees a search history. Display result and home screen. That page contains a stream graph, files, authors, branches, etc.
- **Main Success Scenario**
  1. User goes to the webapp.
  2. Searches for a public repository
- **Extensions**
  2a) User searches for a private repository and only sees it if logged in with appropriate access
- **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
- **Technology and Data Variation List**
  Browser has to support our javascript libraries
  Webapp has to support user login for extension 2a)
- **Frequency of Occurrence**
  Every time they open the program to use it. Once it's already open, they just have to search history.

Use Case 2: Clicking on a repository, user gets lists of files and commits, and a graph of commits by file

- **Stakeholders and Interests**
  People that are interested seeing list of files and commits. Users could be researchers interested in github history of a file, commits and graph of commits by file.
- **Preconditions**
  Has repository displayed
- **Success Guarantee**
  User gets full list of files and commits. Also, on the side bar graph displayed when user clicked file
- **Main Success Scenario**
  1) User sees graphs of information they want for a given file
  2) User sees the list of files by commits
  3) User sees the commits
  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**
  1a) Can also download information
  2)) 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**
  Only when people care about displaying in a presentation

Use Case 3: Once in a repository, user changes what parameters the commits are grouped by (authors or branches)

- **Stakeholders and Interests**
  People that are interested seeing commits grouped by authors or branches
- **Preconditions**
  Internet is on, browser supports js programs, github isn't down
  Properly displayed home screen, list of commits, etc.
- **Success Guarantee**
  User sees a page with tabs (authors, branches or files)to filter. That page contains a stream graph, files, authors, branches, etc.
- **Main Success Scenario**
  1. All repository displayed in the home screen
  2. User clicked authors or branches
  3. Can click on the branches to see commits grouped by branches
  4. Can click on authors to see commits grouped by authors
- **Extensions**
  1. User searches for a private repository and only sees it if logged in with appropriate access
  2. Filter based on files/folder
     a) Can click on one to view commits affecting that file
     b) Can click on the commit to view details for that commit
  3) 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
- **Technology and Data Variation List**
  Browser has to support our javascript libraries
  Webapp has to support user login for extension 2a)
- **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 4: User adds a regex filter for file, author, or branch, eliminating results

- **Stakeholders and Interests**
  People that are interested seeing filter results according to file, author, or branches.
- **Preconditions**
  Internet is on, browser supports js programs, github isn't down
  Properly displayed home screen, list of commits, etc.
- **Success Guarantee**
  User sees a page with tabs (authors, branches or files)to filter.
  That page contains a stream graph, files, authors, branches, etc.

- **Main Success Scenario**
  1. All repository displayed in the home screen
  2. User clicked files, authors or branches
  3. Display results based on file, or author, or branch. Also displays graph for each result
- **Extensions**
  2a) Click on each commits to see the message

- **Special Requirements**
  Boxes containing information scroll properly when the list is longer than the box
- **Technology and Data Variation List**
  Browser has to support our javascript libraries
- **Frequency of Occurrence**
  Every time user click on desired filter like file, author or branch

Use Case 5: User imports a saved history by uploading a file

- **Stakeholders and Interests**
  Users interested in restoring a session previously saved to disk
- **Preconditions**
  User possesses a file containing a previously saved visualization session
- **Success Guarantee**
  User sees the state of the repo as it was seen when saved, and can manipulate as before
- **Main Success Scenario**
  1. User selects "import session from file"
  2. User is presented a file upload dialog
  3. User selects the file containing the session
  4. Session is restored to the same appearance and functionality as when it was saved
- **Extensions**
  4a) If the state of the repo has changed, show a small banner, informing and offering to reload
  4b) User can still add/remove filters and view different aspects of the repository
- **Special Requirements**
  Informative banner is an overlay and doesn't cause the page to annoyingly scroll
  Successive releases of the webapp are backwards compatible if file format changes
- **Technology and Data Variation List**
  User's browser must authorize file uploads
- **Frequency of Occurrence**
  As often as user needs to load from a file

Use Case 6: User adds a date restriction, eliminating results and confining graph

- **Stakeholders and Interests**
  Users interested in seeing only commits within a certain date range
- **Preconditions**
  User has loaded a repository
  Repository has existed for at least one day

- **Success Guarantee**
  No commits are displayed outside the selected range, and the range bounds the graph
- **Main Success Scenario**
  1. User enters a past date into the “start date” or “end date” field
  2. The commit list changes to only contain commits within this range
  3. The graph changes to be bounded by this date range
- **Extensions**
  1a) User enters an end date before the start date: change the end date to the start date
  1b) User enters a start date before the repo’s creation date: change it to the creation date
  1c) User enters an end date in the future: change it to the current date
- **Special Requirements**
  Graph’s start and end dates are always clearly displayed
  Date fields are accessible, with a popup to facilitate date selection instead of just string entry
- **Technology and Data Variation List**
  If a user’s browser doesn’t support d3.js, requirements involving graph may be omitted.
  If user is in a different timezone: reflect the repo’s date, not the user’s local date
- **Frequency of Occurrence**
  As often as the user is interested in filtering commits by date

Use Case 7: User selects “volume” or “count” to change the appearance of the F/A/B graph

- **Stakeholders and Interests**
  Users curious about the relationship of commit volume to commit quantity over time
- **Preconditions**
  User has loaded a repository
  Repository has at least one commit
- **Success Guarantee**
  Graph’s stream width now reflects total commit size in bytes, not commit count
- **Main Success Scenario**
  1. User selects “volume” to change from the default “count” graph display mode
  2. Streamgraph changes so that stream width reflects total commit size in bytes, not commit count
- **Extensions**
  1a) User selects “count” to change back to the default mode, and the graph is reverted
- **Special Requirements**
  If all commits have the same size in bytes, the graph will not visibly change
- **Technology and Data Variation List**
  User’s browser must be compatible with d3.js
- **Frequency of Occurrence**
  As often as the user wants to change the display mode of the graph

Use Case 8: User selects a file/author/branch from the list (lower left), adding/removing a filter for it

- **Stakeholders and Interests**

  Users interested in narrowing down the displayed commits according to their files, authors, or branches (F/A/B), depending on which mode is selected (c/f UC3)

- **Preconditions**
  User has loaded a repository
  Repository contains at least one commit
- **Success Guarantee**
  A filter has been added to the filter box corresponding to the selected F/A/B, and only commits matching this criterion are displayed in the commit list
- **Main Success Scenario**
  1. User selects a F/A/B from the list
  2. A filter is added to the box corresponding to the selected F/A/B
  3. Only commits matching this criterion are displayed in the commit list
- **Extensions**
  2a. Clicking on an x at the filter removes it from the list of things to filter on
- **Special Requirements**
  The selected F/A/B in the list is visibly highlighted
  The selected F/A/B's stream in the graph is highlighted
- **Technology and Data Variation List**
  If user's computer has limited graphical capabilities, highlighting may be omitted.
- **Frequency of Occurrence**
  As often as a user wants to quickly add a single-F/A/B filter

Use Case 9: User selects a commit from the list (middle left), seeing detailed commit info

- **Stakeholders and Interests**
  People interested in information about a specific commit
- **Preconditions**
  Has successfully found a repo and gotten to a file/author/branch to see the list of commits
- **Success Guarantee**
  The bottom right window changes from showing files to showing detailed information about the commit. This includes who committed it, the date, perhaps a list of changes, etc. The graph changes to display the size of files relative to the commit, instead of vice versa
- **Main Success Scenario**
  1. All files/authors/branches/etc. are shown in the lower left
  2. User clicks on a specific file/author/branch
  3. User clicks on a specific commit located in the bottom middle window and sees detailed information about that commit
- **Extensions**
  2a) User can apply filters to refine their search in this step
- **Special Requirements**
  The graph must change to show data about a commit's files, instead of file's commits
- **Technology and Data Variation List**
  Can still download this page even filtered off of commits
- **Frequency of Occurrence**
  Whenever the user is interested in a commit more than a file/author/branch.

Use Case 10: User clicks one of the stacked graph elements, adding an associated filter to the box

- **Stakeholders and Interests**
  People who want to filter but don't want to type the filter out (or see an interested trend and want to investigate, etc)
- **Preconditions**
  Graph is properly loaded and displaying data. Each (color, etc) of the graph is clickable.
- **Success Guarantee**
  A new filter is applied to the data. Most of the things on the screen update.
- **Main Success Scenario**
  1) User sees graph filled out based on some criteria filtered in some manner
  2) User clicks on an element of the graph
  3) Program then applies the graph element as a filter (based on whatever they clicked)
- **Extensions**
  2a) Clicking on an element of the table related to file size doesn't do anything (we have no filter related to file size)
- **Special Requirements**
  The graph is displayed with HTML/etc. elements that can be clicked, rather than just a flat image
- **Technology and Data Variation List**

- **Frequency of Occurrence**
  Probably fairly frequently. Whenever the user wants to learn more and go more in depth about whatever is currently filtered.

Use Case 11: Repo state changes while user is viewing something, and the view updates

- **Stakeholders and Interests**
  Every user of this program
- **Preconditions**
  The program is loaded in the window and the user is browsing it
- **Success Guarantee**
  If something changes in the repository underneath the user, it live-updates and refreshes the views on the page
- **Main Success Scenario**
  1) User is browsing the repo in the program
  2) User (or a different user) makes a change to the repository
  3) View updates so the user isn't looking at stale data
- **Extensions**
  3a) could pop up a banner instead notifying the user of changes and asking to update
- **Special Requirements**
  Program must be able to gracefully handle things like the repo being deleted, massively changed, etc. Example what happens if the file the user is inspecting gets deleted.
- **Technology and Data Variation List**
  Program must be live-querying Github every few seconds in order to live-update
- **Frequency of Occurrence**
  As frequently as the repository is being used. In everyday use, probably not much.

Use Case 12: User views a file/repo that's been deleted, and the view provides a notification

- **Stakeholders and Interests**

Any user

- **Preconditions**
  A few. First, the view is properly rendering with files to be clicked on. Secondly, the above use case (#11) isn't actually implemented. If it were live updating, this wouldn't be a possibility!
- **Success Guarantee**
  The user is shown a page or modal dialogue notifying them that what they clicked on is no longer valid.
- **Main Success Scenario**
  1) User is on a valid page
  2) User clicks on an element of the page (branch, file, etc.) that has been deleted since he loaded the page
  3) User is redirected to a page with a "file/branch/etc. deleted" message
- **Extensions**
  2) Alternatively, the page could live-update so this isn't possible
  3a) Instead of loading a new page, it could instead just open a modal dialogue
- **Special Requirements**
  The page must not live-update
- **Technology and Data Variation List**
  Page cannot be relying on cached data to do its displays (it must query Github every time in order to detect a deleted file)
- **Frequency of Occurrence**
  Very rare. Shouldn't affect most users very often.

Use Case 13: 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

Use Case 14: Exporting the history of a file/author/branch to a url

- **Stakeholders and Interests**
  People that are interested in using the data that is displayed and perhaps sharing it with a friend, on twitter, or elsewhere on the internet
- **Preconditions**
  Has properly found the file or filtered by whatever and viewed the visualizations they want to download
- **Success Guarantee**
  Our server hosts the file online and the user is provided a link to access it
- **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 link to a page containing all the info they want
  3) Can share the link and distribute it as they please
- **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
  The link needs to save the whole page. Whether it can continue to be modified or not is up to implementation.
- **Frequency of Occurrence**
  Probably fairly rarely. When a user wants to share things with friends perhaps, publicizing reports on a repo, etc.

Use Case 15: **User authenticates via username/password & selects a private repository**

- **Stakeholders and Interests**
  User who has login credentials and interested in private repository
- **Preconditions**
  Internet is on, browser supports js programs, github isn't down.
  Should/must have username and password to selects private repository.
- **Success Guarantee**
  User able to login
  User sees a search history and all private repository
  Web App home screen that page contains a stream graph, files, authors, branches, etc.
- **Main Success Scenario**
  1. User goes to the webapp.
  2. Login in to the GitHub
  3. Searches for a public repository
  4. Searches for private repository
  5. Displays Web App home screen that page contains a stream graph, files, authors, branches, etc

- **Extensions**
  User searches for private commits by file, authors or branches.
- **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
- **Technology and Data Variation List**
  Browser has to support our javascript libraries
  Webapp has to support user login for extension
- **Frequency of Occurrence**
  Every time a user with login credentials login

Use Case 16: User selects a different graph type

- **Stakeholders and Interests**
  Any user
- **Preconditions**
  User has selected a repository
- **Success Guarantee**
  The graph is changed from a streamgraph to a bar or line graph
- **Main Success Scenario**
  1. User selects the "change graph type" button in the lower left of the streamgraph
  2. In the resulting popup, user selects the desired type of graph.
  3. Graph changes to reflect the selected type (bar, line, or stream)
- **Extensions**
  2a) The user can select high-contrast or color-blind-friendly display modes for the graph.
- **Special Requirements**
  The colors in the stream/bar/line graphs are appropriately chosen for good visibility.
- **Technology and Data Variation List**
  If d3.js is not available and a fallback is being used, not all of the options may be available.
- **Frequency of Occurrence**
  As often as the user decides that streamgraphs aren't the optimal visualization tool.