User Experience (UX) Guide Artifact Management using GitHub

Cona edits on 25sep20202 … these edits provide flexibility for Teams > BAC files to be organized in a more flexible manner, and redefines UX Guide Structure to contain files that are ready for HFE review.

The purpose of this document is to describe the GitHub repository that enables staff, contractors, and partners to collaborate on content development efforts for the User Experience (UX) Guide for Veterans Health Administration (VHA) Human Factors Engineering Team. The repository structure is outlined in Section I for tracking project-related information and status. Section II describes the naming conventions for artifacts as they are added to the repository and taken through the editorial process. Section III provides information about the GitHub platform and how to perform helpful functions such as creating and downloading a repository, creating a new folder, and uploading/downloading files.

# I. Repository Structure

For reference, the UX Guide GitHub repository currently appears at:

<https://github.com/neugenevcp/VHA-UX-Guide>

It is important to note that this link will change when ownership of the repository is transferred and maintain the following format:

https://github.com/*ADMIN-USERNAME*/VHA-UX-Guide

The repository currently contains the following components:

1. README
2. Teams folder
3. UX Guide Structure folder
4. Protype folder

The following sections describe the expected content for each component.

## README.md

The *README.md* will contain summary information about the project. This file should outline the project goal, identify stakeholders and users and their roles in the project, and an explanation of how to use the repository. The following is an example outline for the README file.

1. Project Goal
2. Stakeholders and Roles
3. Workflow for Uploading Draft Content
4. UX Guide File Naming Conventions

As stated, the Project Goal of the repository is to store content for the UX Guide created by contributing partners through the editorial process prior to final approval by VA. It is also used for issue tracking and version control across the platform and teams building the UX Guide.

The Stakeholders and Roles information should identify the names and roles of the people involved in the project. For teams composed of multiple organizations, identification of the organization of each participant is helpful. For this project, the members and leads of the VHA HFE Team, members of the UX Guide Team, contractors and members of the Project Team, and any other contributors and collaborators should be identified. The owner of the GitHub Repository may also be specified in this section.

It may also be helpful to outline workflow processes for populating the repository in this section as well as file naming conventions for efficient use this tool. The README.md file should be kept up to date as processes evolve over the course of the project.

Here is an example of the main page of the repository:

A screenshot of a computer

Description automatically generated

## Teams

When adding new content to the repository, users should start this process in the “**Teams”** folder. The “**Teams”** folder is structured by Organization or Team and then typically by Project within each Team. There are typically separate folders for draft content and finalized content within each project folder. Here are examples of the “**Teams”** folder and subsequent structures:

A screenshot of a computer screen

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer screen

Description automatically generated

**Workflow for Uploading Draft Content**

The workflow for entering draft content in the repository begins in the **“Teams”** folder. To begin uploading draft content, go to the "**Teams"** folder, select the appropriate team, and then select the appropriate **“Project”** folder. Add the file to the **"Draft"** folder. Drafts will be edited within this folder using the file naming conventions as described under *Section II. UX Guide Naming Conventions Guidance*.

After changes have been incorporated from all versions, content will be labeled “ready for review” and added to the **“Final – To Be Approved”** folder for review by the VHA HFE Team. Once approved, the content file status will be labeled “approved” and will be promoted to the UX Guide Structure in its respective folder.

## UX Guide Structure

The **UX Guide Structure** folder will serve as the last step in the content development process, as it will house the ~~finalized~~ content ready for review by HFE. The structure currently mirrors that of the UX Guide site with the following folders for corresponding content: **Home**, **Fundamental Concepts**, **UX Process**, **Methods**, **Tutorials**, **Resources**.

It is helpful to note the **Methods** folder content will describe UX methods, skills, and activities used in human-centered research and design, and their applications to improve usability and user satisfaction. Examples of content topics in the **Methods** folder include Clinical Workflow Modeling, Heuristic Evaluation, Stakeholder Interview, Usability Walkthrough, User Interview, and Visual Modeling.

The **Tutorials** folder will store content pertaining to comprehensive tutorials developed by UX experts to assist end users in learning a range of human-centered design skills (i.e.—Workflow Modeling).

The **Resources** folder will store a variety of content namely, case studies, design examples and guidelines, FAQs, newsletters and publications, personas, playbooks, quick start guides, tools (i.e.—checklists and templates), and videos.

It is important to note that the **UX Guide Structure** folder is evolving and subject to change to address project needs.

## Prototype

The **Prototype** folder will house miscellaneous assets for the UX Guide site in the prototype phase, and its uses will evolve based on the project needs of the UX Guide Team.

# II. UX Guide Naming Conventions Guidance

The following naming conventions have been developed by the UX Guide Team to promote efficient workflow and organization within the repository:

File name = content-type\_descriptor \_version-revision (status)

**Content-type**

Content-type would be a label used for the content such as: “case-study” or “video-script”.

**Descriptor**

The descriptor should give specific project information about the file. For example, “Formative Usability Tests for Design”.

**Version**

To provide added version control during the editorial process the following numbering convention should be used:

* v1 = first version, before any external review or user feedback
* v2 = second version, incorporating feedback from subject matter experts (SMEs) or users, or reflecting any changes after v1 has gone for review or been finalized.
* v3 = third version, reflecting any changes after v2 has gone for review or been finalized.

**Revision**

With each revision within a version — including edits from SMEs, edits addressing feedback from end users, or other edits — the file number goes up one. For example:

* v1-01
* v1-02
* v1-03

**Status**

The file status can be:

* In progress = the content is being drafted or edited
* Ready for review = file is ready for SME review
* Approved = content has been approved by HFE team

**Examples**

* case-study\_Formative Usability Tests for Design Solution\_v1-01 (in progress)
* video-script\_Common Usability Issues for Clinical Reminders\_v2-02 (approved)

# III. General GitHub Information

This section provides some information that is useful for any repository. It provides pointers to helpful documentation provided on the GitHub “docs.GitHub.com” site.

## Creating a GitHub Account

To view and download content from a public GitHub repository, no user account is needed. However, a GitHub account is needed to upload content and edit the repository. GitHub accounts range from free, basic accounts for teams and developers to more complex enterprise accounts for a monthly fee.  Visit the following sites for more information on setting up a GitHub Account and getting started with GitHub:

<https://docs.github.com/en/github/getting-started-with-github>

[https://www.toolsqa.com/git/how-to-create-GitHub-account/](https://www.toolsqa.com/git/how-to-create-github-account/)

## Importing a Repository

If you would like to create a new repository based on an existing one, you can use the importing function to get started.

[https://docs.GitHub.com/en/GitHub/importing-your-projects-to-GitHub/importing-a-repository-with-GitHub-importer](https://docs.github.com/en/github/importing-your-projects-to-github/importing-a-repository-with-github-importer)

## Downloading a Repository

The  icon allows you to download a zip file of an existing repository. You will find this icon at the top right of a repository on the main page. Once selected, you will see the option to select “Download zip” that will create a zip file and start a download of the repository files. An advantage of this feature is that it will create a version of each of the text documents. Here is an example of the main page of a repository:

A screenshot of a social media post

Description automatically generated

## Creating a Repository

To share files and collaborate within a project, a repository can be created from scratch. To create files and folders for your structure:

1. Go to  next to .
2. Select “Create new file”.
3. Enter the name of the file into the empty cell.
4. Commit the changes. To confirm changes in a GitHub, users must *commit* them. A commit is an individual change made to a file or folder. Each commit has a unique ID, called a *hash*, which allows users to keep a record of specific changes that were made to the repository. This may be useful if a user needs to revert the repository back to a specific point in time to undo changes. When committing changes, collaborators can add a brief description of each change that was made (i.e.— “Created index.html file.”) To confirm any change, click the  icon at the bottom left.

Here is an example page when creating a new file:

A screenshot of a cell phone

Description automatically generated

Folders can be created following the same process:

1. Go to  next to .
2. Select “Create new file”.
3. Enter the name of the file into the empty cell.
4. Enter “/” and a new cell will appear.
5. Enter a file name to finalize your folder. A folder cannot exist in the repository without having at least one file enclosed. As a placeholder, “index.html” or “deleteme.md” can be used.
6. Click the  icon at the bottom left.

Similarly, these files and folders can be deleted directly on GitHub by following these steps:

1. Select the file you wish to delete
2. Click theicon, and a new page will load to confirm the changes to commit.
3. A default commit description will be provided; however, the commit description may be edited if needed.
4. Click the  icon.

Here is an example page when a file is selected:

A screenshot of a cell phone

Description automatically generated

Here is an example page when deleting a file:

A screenshot of a cell phone

Description automatically generated

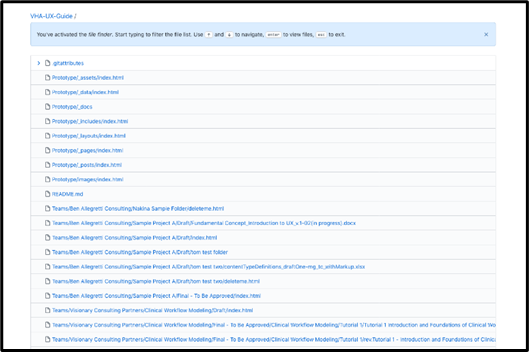
Folders can be deleted within GitHub by deleting all enclosed files, as empty folders cannot exist in a repository.

Visit the following site for further step-by-step instructions for creating a repository, naming, and setting public, internal, or private visibility:

[https://docs.GitHub.com/en/GitHub/getting-started-with-GitHub/create-a-repo](https://docs.github.com/en/github/getting-started-with-github/create-a-repo)

## Viewing all files

Once your repository is created, files can be viewed by clicking the  icon on the main page. Once selecting it, you see the *file finder* view of the files in the repository in alphabetical order. Here is an example of the *file finder* view:



## Uploading and Downloading Files/Folders to a Repository

Once your repository is created, code files and folders can be created directly in GitHub or uploaded to organize the structure. Non-code files, such as Word documents and images, can be uploaded to GitHub as well. To upload files:

1. Go to  next to .
2. Select “Upload files” and a new page will load.
3. Files can be dragged and dropped directly into the repository directory. Files can also be selected by clicking the “choose your files” link.

Here is an example page when “Upload files” is selected:

A screenshot of a social media post

Description automatically generated

Folders containing content can also be dragged and dropped into the repository. Empty folders cannot be added. The folder must contain at least one file to be added to the repository. Placeholder files such as “index.html” or “deleteme.md” can be used to set up folders for the repository structure before artifacts are added. Note that files larger than 25 MB cannot be uploaded using GitHub. To upload larger files, refer to *Section J. Uploading Files Larger than 100MB to a Repository*.

Code files created in GitHub can be renamed, moved, edited, and deleted within the GitHub platform. Visit the following site for more detailed instructions on managing files in a GitHub repository:

[https://docs.GitHub.com/en/GitHub/managing-files-in-a-repository/managing-files-on-GitHub](https://docs.github.com/en/github/managing-files-in-a-repository/managing-files-on-github)

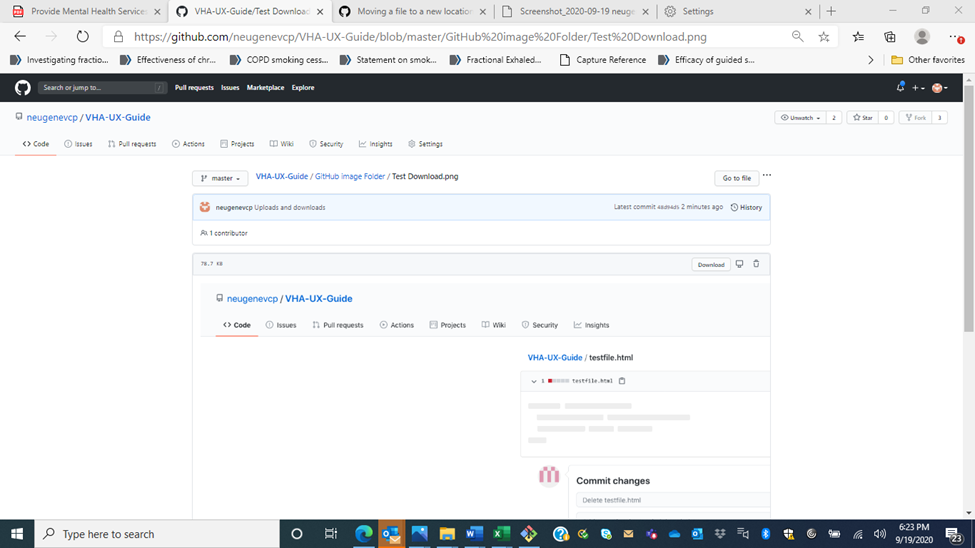
For non-code files uploaded to GitHub, these changes must be done locally using the command line. Refer to *Section H. Working Locally in a Repository* for more information on managing non-code files, andvisit the following site for more detailed instructions on managing files using the command line:

<https://docs.github.com/en/github/managing-files-in-a-repository/moving-a-file-to-a-new-location-using-the-command-line>

To download files from GitHub:

1. Select the file that you wish to download by clicking the  icon in the top right or by navigating the repository.
2. Select the  icon in the top right of the file viewer window.

Here is an example page when a non-code file is selected:



## Adding Collaborators

A collaborator is a user with read and write access to a repository. To upload files and make changes to the repository, a collaborator must first be invited to contribute. With a GitHub free account, unlimited collaborators can be added to public repositories and up to three collaborators to a private repository owned by a personal account. Upgrading to GitHub Pro will allow users to add more collaborators. Visit the following site for more information on adding and removing collaborators and managing access to your repository:

[https://docs.GitHub.com/en/GitHub/setting-up-and-managing-your-GitHub-user-account/managing-access-to-your-user-accounts-project-boards#inviting-collaborators-to-a-user-owned-project-board](https://docs.github.com/en/github/setting-up-and-managing-your-github-user-account/managing-access-to-your-user-accounts-project-boards#inviting-collaborators-to-a-user-owned-project-board)

## Working Locally in a Repository

While some files can be managed directly on GitHub, non-code files, such as images, require management locally on your computer, and changes can be pushed to GitHub using the command line. To begin, git must be downloaded. Once git is installed, commands can be entered in a terminal through the Git Bash app, a program included in the git installation. Visit the following site to download and install git for Windows or MacOS:

<https://git-scm.com/download/>

**GitHub Desktop**

GitHub offers a desktop app as a tool that allows users to interact with GitHub locally. To download GitHub Desktop, visit the following site:

<https://desktop.GitHub.com/>

Once GitHub Desktop is installed and launched, sign in using your GitHub credentials. Next, add a repository to your GitHub Desktop to start collaborating. New repositories can be created in the hard drive. Existing repositories can also be added from the hard drive or cloned from GitHub online. To use the command line using GitHub Desktop, downloading git is still required. Once git is installed, the command line can be accessed by selecting “Open in Command Prompt” under the Repository Tab.

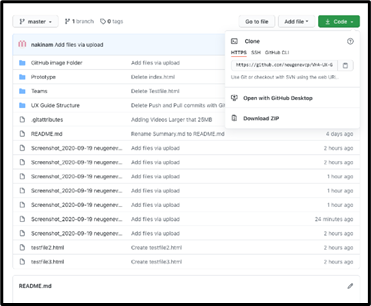
To collaborate with your team, the existing GitHub repository must be cloned to your computer. Visit the following site for step-by-step instructions in setting up GitHub Desktop:

<https://www.sqlservercentral.com/blogs/setting-up-github-desktop-formerly-github-for-windows>

**Cloning a Repository**

Cloning an existing GitHub repository will create a full copy on the user’s computer, allowing the user to make edits, add, and remove files, and push larger commits locally and easily. Changes can be made to the repository directory folders and pushed to the GitHub repository using the command line. To clone an existing repository:

1. On the main page of the GitHub repository, click the  icon at the top right of the window.
2. Select Download ZIP, Open in GitHub Desktop, or click the  icon to copy the URL to enter into the command line. See the following for an example:



1. To enter using the command line, open a terminal through the Git Bash app.
2. Type “git clone” and then the paste the URL from GitHub:

$ git clone https://github.com/*YOUR-USERNAME*/*YOUR-REPOSITORY*

1. Press Enter to create the clone.

Visit the following site for more information on cloning repositories:

<https://docs.github.com/en/github/creating-cloning-and-archiving-repositories/cloning-a-repository>”

**Git Commands**

Git allows users to continue making changes to the local directory. The “git pull” command allows users to pull updates from the remote repository on GitHub. The “git status” command can detect changes made to the local repository that are not on the GitHub instance. Once files are added or edited, they can be tracked using the “git add filename” command. As with GitHub online, any changes must be committed before they are saved to the repository. Using the “git commit -m ‘descriptive message’” command, commits can be described and made, and then, pushed to the remote repository using the “git push” command. Users will need read and write access to the remote repository on GitHub to push changes. Visit the following sites for step-by-step instructions on using the command line to manage your local repository in addition to basic git commands with examples:

[https://docs.GitHub.com/en/GitHub/managing-files-in-a-repository/adding-a-file-to-a-repository-using-the-command-line](https://docs.github.com/en/github/managing-files-in-a-repository/adding-a-file-to-a-repository-using-the-command-line)

<https://rubygarage.org/blog/most-basic-git-commands-with-examples>

## Tagging

Users can tag specific points in a repository’s history to denote importance. Tags can be used as a pointer for specific commits (lightweight) or stored as full objects in the Git database, containing useful information such as the tagger’s name, email, date, and tagging message (annotated). Creating tags can be useful in the version control process. Visit the following site for more information on creating and deleting different types of tags:

<https://git-scm.com/book/en/v2/Git-Basics-Tagging>

## Uploading Files Larger than 100MB to a Repository

GitHub limits the size of files allowed in any repository. Files that are added via a browser are limited to 25 MB per file. Using the GitHub desktop app, files up to 100 MB each can be added. Users will receive a warning when uploading files over 50 MB, and the platform blocks pushes that exceed 100 MB. To upload files larger than 100MB, users will need to download and install Git Large File Storage (Git LFS). Visit the following site to download Git LFS and to access step-by-step instructions and commands:

[https://git-lfs.GitHub.com/](https://git-lfs.github.com/)

Every user with Git LFS enabled will begin with 1GB of storage and 1 GB per month of free bandwidth. Upgrades in bandwidth and storage quotas are available for a monthly fee. Visit the following sites to access step-by-step instructions on the process and retrieve pricing information:

[https://docs.GitHub.com/en/GitHub/managing-large-files](https://docs.github.com/en/github/managing-large-files)

[https://GitHub.community/t/what-are-the-pricing-plans-for-git-lfs-is-there-something-better-than-git-lfs-to-use/519](https://github.community/t/what-are-the-pricing-plans-for-git-lfs-is-there-something-better-than-git-lfs-to-use/519)

## Branches

Branches help the team separate work under development without affecting other branches in the repository. Each repository has one default branch that GitHub displays when anyone visits the repository. Multiple users can have other branches. The team can merge a branch into another branch using a pull request. Visit the following site to read more about the use of branches:

[https://docs.GitHub.com/en/GitHub/collaborating-with-issues-and-pull-requests/about-branches](https://docs.github.com/en/github/collaborating-with-issues-and-pull-requests/about-branches)

## Resetting a Repository

It may be useful for a user to revert to a previous version of the repository should unexpected errors or insurmountable hindrances occur. This process is straightforward using the command line. Visit the following sites to learn more about this process:

<https://opensource.com/article/18/6/git-reset-revert-rebase-commands>

<https://youtu.be/FdZecVxzJbk>