**TASCore Design   
Style Guide**

Version 2.0

Standards, Design Principles, Templates, UI,   
Styles and Branding



January 2019

Department of Veterans Affairs

Office of Information and Technology (OI&T)

**Revision History**

| Date | Version | Description | Author |
| --- | --- | --- | --- |
| 2018-01-04 | 2.0 | Tableau reporting | Douglas Weinberg |
| 2018-11-10 | 1.3 | Updating data table | Douglas Weinberg |
| 2018-09-05 | 1.2 | Adding Wireframe Appendix | Douglas Weinberg |
| 2018-09-02 | 1.1 | Added Shared Components Section 3.24; page 40  Added Modal Window  Section 3.23, page 37 | Douglas Weinberg |
| 2017-12-28 | 1.0 | Draft 0.1 | Douglas Weinberg |

**Artifact Rationale**

The TASCore Design Style Guide is a set of guidelines to help designers and developers build trustworthy, accessible, and consistent government websites. This set of standards will allow agencies to quickly prototype and deploy digital products using a baseline of patterns applicable to TASCore project. TAS is an acronym for Transaction Application Suite.

The MCCF EDI TAS portal website was created using a conglomerate of government web standards. Design guides from several existing sources include the VA Playbook, U.S Web Design Standards (USWDS) and 18F. These standards are used to design systems for government websites to provide a more rich, robust user experience.

MCCF TASCore style guide has implemented U.S. Web Design Standards as the foundation style for all user interface (UI) component libraries so that the MCCF TAS website will have a similar and common design among other federal portals.

Table of Contents

[1 Introduction 1](#_Toc534722321)

[1.1 Purpose 1](#_Toc534722322)

[1.2 Goal 1](#_Toc534722323)

[1.3 Background 1](#_Toc534722324)

[1.4 Mission 1](#_Toc534722325)

[1.5 Dependencies 2](#_Toc534722326)

[1.6 Accessibility 2](#_Toc534722327)

[1.7 Standards 2](#_Toc534722328)

[1.8 Design Principles 2](#_Toc534722329)

[2 How to Use this Guide 2](#_Toc534722330)

[2.1 Developers and Designers 3](#_Toc534722331)

[2.2 How do I add a new CSS file? 4](#_Toc534722332)

[2.3 How do I add a JavaScript file? 5](#_Toc534722333)

[2.4 How do I test my changes? 5](#_Toc534722334)

[3 UI Components 5](#_Toc534722335)

[3.1 PrimeNG Components 7](#_Toc534722336)

[3.2 Typography 7](#_Toc534722337)

[3.3 Pairing and Styles 9](#_Toc534722338)

[3.4 Typesetting 10](#_Toc534722339)

[3.5 Links 11](#_Toc534722340)

[3.6 Lists 11](#_Toc534722341)

[3.7 Color Palette 12](#_Toc534722342)

[3.8 Text Accessibility 13](#_Toc534722343)

[3.9 Headers 14](#_Toc534722344)

[3.10 Accessibility 15](#_Toc534722345)

[3.11 Usability 15](#_Toc534722346)

[3.12 Header and TAS custom styling 16](#_Toc534722347)

[3.13 Footers 16](#_Toc534722348)

[3.14 Accessibility 17](#_Toc534722349)

[3.15 Usability 17](#_Toc534722350)

[3.16 Side Navigation 17](#_Toc534722351)

[3.17 Accessibility 18](#_Toc534722352)

[3.18 Usability 18](#_Toc534722353)

[3.19 Expand/Collapse Side Navigation 18](#_Toc534722354)

[3.20 Grids 19](#_Toc534722355)

[3.21 Accessibility 20](#_Toc534722356)

[3.22 Usability 20](#_Toc534722357)

[3.23 Buttons 20](#_Toc534722358)

[3.24 Accessibility 22](#_Toc534722359)

[3.25 Usability 22](#_Toc534722360)

[3.26 Labels 22](#_Toc534722361)

[3.27 Accessibility 22](#_Toc534722362)

[3.28 Usability 22](#_Toc534722363)

[3.29 Data Tables 22](#_Toc534722364)

[3.30 Accessibility 24](#_Toc534722365)

[3.31 Usability 24](#_Toc534722366)

[3.32 Alerts 24](#_Toc534722367)

[3.33 Accordions 26](#_Toc534722368)

[3.34 Accessibility 26](#_Toc534722369)

[3.35 Usability 26](#_Toc534722370)

[3.36 Form Controls 27](#_Toc534722371)

[3.37 Accessibility 27](#_Toc534722372)

[3.38 Text Input 28](#_Toc534722373)

[3.39 Accessibility 28](#_Toc534722374)

[3.40 Usability 28](#_Toc534722375)

[3.41 Dropdown 28](#_Toc534722376)

[3.42 Accessibility 29](#_Toc534722377)

[3.43 Usability 29](#_Toc534722378)

[3.44 Checkboxes 29](#_Toc534722379)

[3.45 Accessibility 30](#_Toc534722380)

[3.46 Usability 30](#_Toc534722381)

[3.47 Radio buttons 30](#_Toc534722382)

[3.48 Accessibility 30](#_Toc534722383)

[3.49 Usability 31](#_Toc534722384)

[3.50 Date input 31](#_Toc534722385)

[3.51 Accessibility 31](#_Toc534722386)

[3.52 Usability 31](#_Toc534722387)

[3.53 Validation 32](#_Toc534722388)

[3.54 Accessibility 32](#_Toc534722389)

[3.55 Usability 32](#_Toc534722390)

[3.56 Form Templates 32](#_Toc534722391)

[3.57 Accessibility 33](#_Toc534722392)

[3.58 Search Bar 35](#_Toc534722393)

[3.59 Accessibility 36](#_Toc534722394)

[3.60 Breadcrumbs 36](#_Toc534722395)

[3.61 Modal Window/Alert messages 37](#_Toc534722396)

[3.62 Understanding Shared Components and Modules 39](#_Toc534722397)

[3.63 Component File Structure 40](#_Toc534722398)

[3.64 Tree View Component 41](#_Toc534722399)

[3.65 3 state checkboxes 42](#_Toc534722400)

[3.66 EXPAND/COLLASPE TOGGLE  43](#_Toc534722401)

[3.67 DEVELOPER NOTES: 43](#_Toc534722402)

[3.68 Dynamic Select box 47](#_Toc534722403)

[3.69 Date Picker 47](#_Toc534722404)

[3.70 Quick Links 48](#_Toc534722405)

[3.71 Tooltips 48](#_Toc534722406)

[3.72 Pagination 49](#_Toc534722407)

[3.73 Usability 49](#_Toc534722408)

[3.74 Accessibility 49](#_Toc534722409)

[3.75 Tableau Reporting 50](#_Toc534722410)

[3.76 Usability 51](#_Toc534722411)

[3.77 Accessibility 52](#_Toc534722412)

[Appendix A – Wireframes 59](#_Toc534722413)

[B. Template Revision History 70](#_Toc534722414)

# Introduction

This document describes how to apply design and brand user interface (UI) components based on the styles of TASCore. This document is for designers and developers to better understand the design principles to easily implement work on the MCCF TAS framework/system in a consistent manner with a high degree of quality, and a high assurance of successful testing through a test-driven-design method.

## Purpose

The overall goal of TASCore Design Guide is to provide digital services that improve the user experience to obtain information in a useful and productive manner. By incorporating USWDS along the VA Playbook Design standards, TAS portal contains a library of design components to allow developers to quickly provide trustworthy, accessible and consistent digital government services.

## Goal

The goal for TASCore Design Guide is to provide a consistent feel with other federal websites to build trust with users, help people identify official websites, and make services easier to use. The Guide aims to help designers and developers provide digital services that improve the user experience in obtaining information in a useful and productive manner. It describes the user experience (UX), styling, and branding standards of TASCore so ANY web designer/developer/UX-person may understand and apply them to a product, and that all such content is contained within this document.

## Background

In the past, the government has produced many websites created by different teams of developers. Each portal had a different look and feel so there was no consistency within these sites. This became a problem for end users since buttons, navigation, color, fonts and styles would be different. The user experience became more fractured as one navigated through these various portals.

## Mission

By using the same design standards as the VA Playbook, the TASCore Design Guide will help build a single unified digital experience for all veterans.

TASCore portal contains a library of design components that allows developers to quickly provide trustworthy, accessible and consistent digital government services.

* Unifying design standards between VA Playbook and USWDS
* Improving customer experience
* Meeting the customer needs
* Creating a responsive platform so users can view information regardless of screen size
* Providing transparency about methods and processes
* Designing efficient end to end processes to allow information to be processed from external systems

## Dependencies

TASCore styles were derived from USWDS, the VA Playbook Design standards and 18F into one complete design standard. The site incorporates a suite of Angular components called PrimeNG. These components will be discussed throughout this document.

## Accessibility

TASCore style includes Section 508 compliance so every asset meets the highest accessibility. All TASCore users regardless of disability status, can access this site. The accessibility styles are based upon the USWDS so the TASCore is accessible to all users. TASCore is compatible with assistive technology including screen readers for those who have vision impairment. See [Design Accessibility](#Accessibity) for more details.

## Standards

TASCore Style Design guide is a standardized set of UI components which are accessible, responsive, and designed for flexibility regardless of the viewport size. These standards include various elements on a web page including:

* Buttons
* Tables
* Navigation
* Typography
* Forms
* Colors
* Style sheets

## Design Principles

* Understand human need: design for people, rather than VA’s systems
* Assume every visitor is new
* Speak clearly, respectfully, and directly
* Help people reach their goals every time
* Connect with customers, including opportunities for feedback and dialog
* Research, observe, test, and continuously improve
* Measure what matters
* Be device agnostic
* Employ modern development practices – be agile

# How to Use this Guide

When adding or updating new TASCore pages, refer to this guide to make sure you maintain the correct styling and branding of UI components that are 508 compliant. All the current CSS classes have been tested and accepted for 508 accessibility and usability.

## Developers and Designers

TASCore file and folder structure

The UI components are built on a solid HTML foundation so they will render seamlessly across all browsers including IE, Chrome and Firefox. It is a good idea to make sure each browser version is up to date.

Below is a screen capture of the TASCore folder structure:

|  |  |
| --- | --- |
|  | **NOTES**: TAS is using version **1.6.3** of the USWDS framework.  The uswds folder must contain the following subfolders:   * CSS – uswds.css, uswds.min.css * Fonts (Source Sans Pro, Merriweather) * IMG – stores. pngs, svg files * JS – uswds.js, uswds.min.js   The css folder contains TASCore custom CSS files and other CSS files related to PrimeNG Angular controls   * tas-custom.css * fullcalendar.css * quill.snow.css |
|  | Font Awesome is stored inside font-awesome-4.7.0 folder   * CSS – font-awesome.css * Fonts – contains all Font Awesome files (.eot, .svg, .ttf, .woff, .woff2, .otf) |
|  | The primeng folder contains assets required to render these UI controls:   * js – stores PrimeNG JavaScript files * resources – stores images and themes related to the control |

## How do I add a new CSS file?

CSS files can be added to TASCore inside a file called app.component.ts.

|  |  |
| --- | --- |
|  | The app folder contains a suite of folders that pertain to site’s navigational structure. These folders contain ts (typescript), css, js, and html files.  The assets folder contains all the site assets including css, img, js, and ttf font files. |

|  |  |
| --- | --- |
|  | CSS files can be added by updating the component inside the app.component.ts file.  @Component({  selector: 'app-root',  encapsulation: ViewEncapsulation.None,  styleUrls: [  '../assets/css/tas-custom.css',  '../assets/uswds/css/uswds.css',  './app.component.css',  '../assets/font-awesome-4.7.0/css/font-awesome.css',  '../assets/primeng/resources/themes/tas-core/theme.css',  '../assets/primeng/resources/primeng.min.css',  '../assets/css/fullcalendar.css',  '../assets/css/quill.snow.css'  ],  templateUrl: './app.component.html'  }) |

## How do I add a JavaScript file?

JavaScript files are added to the .angular-cli.json file located in the root directory of the site.

|  |  |
| --- | --- |
|  | This json file would need to be updated by appending a new js file under the ‘scripts’ key:value pair.  "scripts": [  "assets/js/datetimeclock.js",  "assets/uswds/js/uswds.min.js",  "assets/js/switch-sub.js",  "assets/js/calendar.js",  "assets/js/calendar.min.js",  "assets/primeng/js/quill.js"  ], |

## How do I test my changes?

Since TASCore was developed using Angular and NodeJS, once you have cloned the repository, you’ll need to install its dependencies:

npm install

To start up the site locally on your machine run the command:

npm start

To view the site go to the URL : <http://localhost:4200> in the web browser.

To test any changes applied, run the command:

npm test

This will ensure that any JavaScript and any CSS source files meet USWDS coding standards.

# UI Components

TASCore style provides a set of UI components that are designed for consistency and simplicity across all VA web sites. The list of components includes all the basic elements of a web page including header, global navigation, footers, fonts, color palettes, search bar, etc. Review each component for more detail.

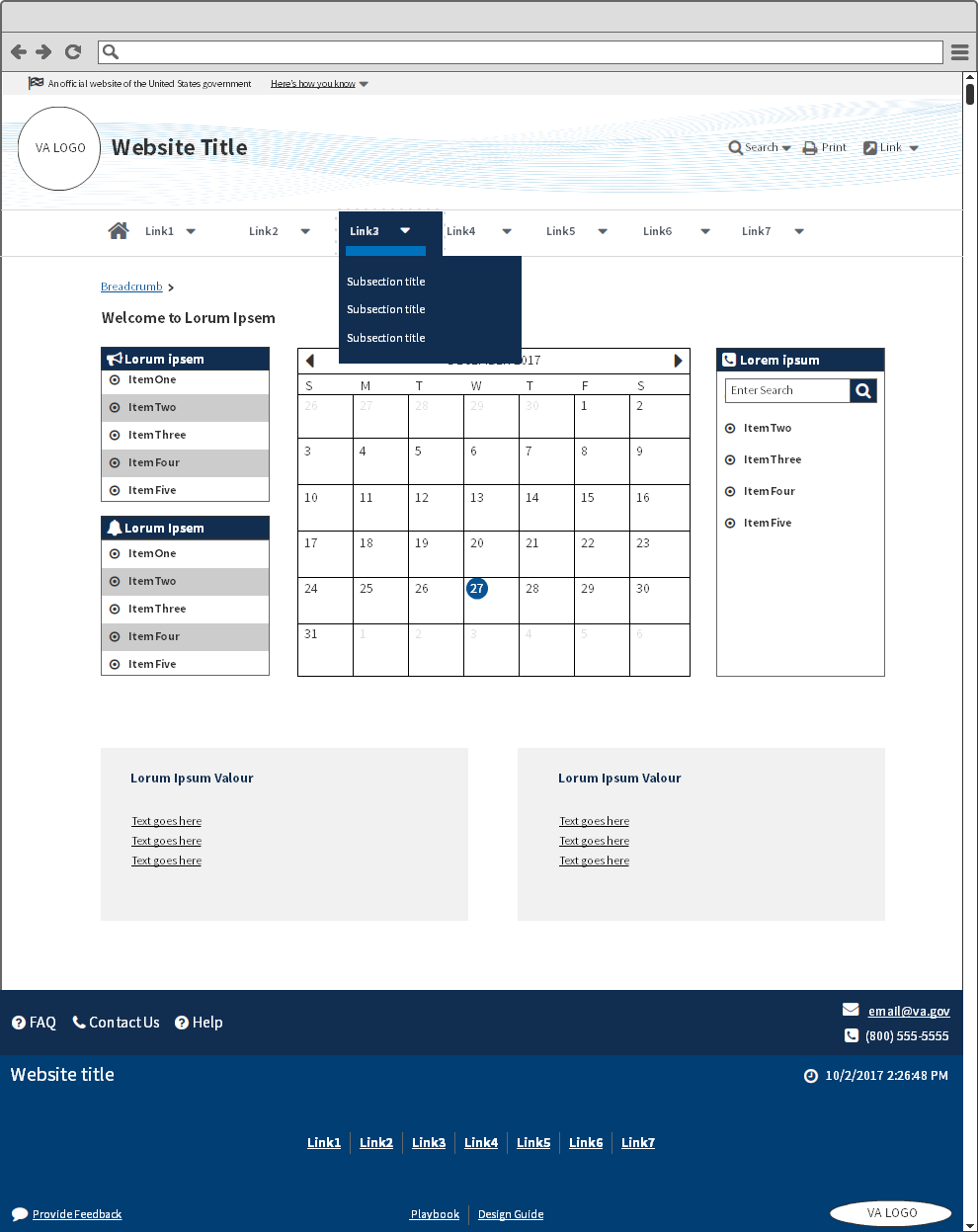


Table 1: List of all UX components for TAS

| ID | Component | Role |
| --- | --- | --- |
| 1 | Typography | Source Sans Pro and Merriweather   * Pairing and Styling * Typesetting * Links * Lists |
| 2 | Color palette | A family of colors used for styling and branding   * Palette * Text Accessibility |
| 3 | Headers | Branding, logos, identification of organization   * Accessibility rules for 508 compliance * Usability |
| 4 | Footers | A set of links at bottom of every webpage   * Footer components |
| 5 | Side Navigation | Vertical navigation to display other web pages on same website   * Single level * Two levels * Three levels |
| 6 | Grids | Responsive Design   * 12 column grid |
| 7 | Buttons | A call to action   * Primary * Secondary |
| 8 | Labels | Draw attention to content   * Style and layout |
| 9 | Tables | Display tabular data in columns and rows   * Style and layout |
| 10 | Alerts/Messages | Keep users informed of specific actions   * Style and layout |
| 11 | Accordions | Expandable and collapsible control   * Style and layout |
| 12 | Form controls | Fields where users can input data   * Style and layout |
| 13 | Form templates | Consistent branding across VA websites   * Style and layout |
| 14 | Search bar | Input fields to allow users to search data based on keywords   * Style and layout |
| 15 | Breadcrumb | Allows visitors keep track of their location |
| 16 | Modal Window | Pop up window for external links |

## PrimeNG Components

[PrimeNG](https://www.primefaces.org/primeng/#/) is a suite of rich UI Angular components used throughout the TASCore site. While developing TASCore and embedding these components in the site, all of them needed to be re-styled according to the USWDS. Since the components have their own sets of CSS files, developers created a new set of custom CSS files, stored in the aptly named tas-custom.css folder.

The tas-custom.css folder contains classes that overwrite the original classes so that the components will adhere to styles and branding of the USWDS. This guide will explain the classes used to restyle any of the Angular components.

## Typography

TASCore uses 3 font families:

* Source Sans Pro - used in body copy
  + Regular, weight 400
  + Bold, weight 700
* Merriweather – used in headings H1 to H5; H6 is Source Sans Pro
  + Regular, weight 400
  + Italic, weight 400
* Font Awesome - icons used in header, footer and body

|  |  |
| --- | --- |
| **Source San Pro – san serif font** | **Merriweather – serif font** |
| <https://fonts.google.com/specimen/Source+Sans+Pro> | <https://fonts.google.com/specimen/Merriweather> |
|  |  |
| **Font Awesome**  <http://fontawesome.io/icons/> | |
|  | |

Font Awesome has its own CSS file linked to the TASCore site called font-awesome.css. This file contains all the .fa classes to display any Font Awesome icon within the suite. The font is declared in the HTML:

<span class="fa fa-commenting-o" aria-hidden="true"></span>

Aria-hidden attribute is used so screen readers skip the font from being read out loud.

|  |  |
| --- | --- |
|  | .fa {  display: inline-block;  font: normal normal normal 14px/1 FontAwesome;  font-size: inherit;  text-rendering: auto;  -webkit-font-smoothing: antialiased;  -moz-osx-font-smoothing: grayscale; } |

## Pairing and Styles

Merriweather headings used with Source Sans Pro body:

|  |  |
| --- | --- |
| **Display** | font-family: ‘Merriweather’ font-weight: 700 font-size: 52px or 5.2rem line-height: 1.3em/68px |
| **Heading 1<h1>** | font-family: ‘Merriweather’ font-weight: 700 font-size: 40px or 4rem line-height: 1.3em/52px |
| **Heading 2 <h2>** | font-family: ‘Merriweather’ font-weight: 700 font-size: 30px or 3rem line-height: 1.3em/39px |
| **Heading 3 <h3>** | * font-family: ‘Merriweather’ font-weight: 700 font-size: 20px or 2rem line-height: 1.3em/26px |
| **Heading 4 <h4>** | font-family: ‘Merriweather’ font-weight: 700 font-size: 17px or 1.7em line-height: 1.3em/22px |
| **Heading 5 <h5>** | font-family: ‘Merriweather’ font-weight: 700 font-size: 15px or 1.5rem line-height: 1.3em/20px |
| HEADING 6 <h6> | font-family: ‘Source Sans Pro’ font-weight: 400 font-size: 13px or 1.3rem line-height: 1.5em text-transform: uppercase |
| Lead paragraph | font-family: ‘Merriweather’ font-weight: 400 font-size: 20px or 2rem line-height: 1.7em/34px |
| Body copy. A series of sentences together which make a paragraph. | font-family: ‘Source Sans Pro’ font-weight: 400 font-size: 17px or 1.7rem line-height: 1.5em/26px |
| Italic body copy. | font-family: ‘Source Sans Pro’ font-style: Italic font-weight: 400 font-size: 17px or 1.7rem line-height: 1.5em/26px |

**NOTE: 1rem = 10px**

## Typesetting

Readable text to allow users to take in textual information.

The Grand Canyon

Grand Canyon National Park is the United States’ 15th oldest national park. Named a UNESCO World Heritage Site in 1979, the park is located in Arizona.  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

75 characters max on desktop. Paragraphs line lengths should be between 50 and 75 characters per line while 66 characters is considered ideal. White spacing used between headers and body copy should be 60px, 30px, 20px or 15px.  
See Code Review Exhibit D

|  |
| --- |
| <H1> Section Heading  .usa-font-lead { Merriweather, font-size:4rem, font-weight:700 } |
| **<H2> Section Heading**  .usa-font-lead { Merriweather, font-size:3rem, font-weight:700 } |
| **<H3> Section Heading 3**  .usa-font-lead { Merriweather, font-size:2rem, font-weight:700 } |
| **<H4> Sub Section Heading 4**  .usa-font-lead { Merriweather, font-size:1.7rem, font-weight:700 } |
| **<H5> Sub Section Heading 5**  .usa-font-lead { Merriweather, font-size:1.5rem, font-weight:700 } |
| <H6> Sub Section Heading 6  .usa-font-lead { Source Sans Pro, font-size:1.3rem, font-weight:400 } |
| <p> Body copy |
| .usa-content p {Source Sans Pro, font-size:1.7, font-weight:400} |

## Links

Hyperlinks are used to navigate to different webpages or websites.

[This](javascript:void(0);) is a text link on a light background.

[This](javascript:void(0);) is an active link. It turns purple on downstate, back to gray on upstate.

This is a link that goes to an [external website](http://media.giphy.com/media/8sgNa77Dvj7tC/giphy.gif) .

[This](javascript:void(0);) is a text link on a dark background.

|  |  |
| --- | --- |
|  | <ul class="usa-unstyled-list usa-nav-secondary-links">  <li>  <a [routerLink]="['/common/faq']" title="Frequently Asked Questions">  <span class="fa fa-commenting-o" aria-hidden="true"></span> FAQ  </a>  </li>  <li>  <a [routerLink]="['/common/contact']" title="Contact Info">  <span class="fa fa-phone-square" aria-hidden="true"></span>Contact Us  </a>  </li>  </ul> |

## Lists

Lists are used to organize information in a structural organization like an outline. They come as either an unordered list or ordered list.

UNORDERED LIST <h6>

* font: Source Sans Pro
* font-size: 1.7rem
* line-height: 1.5rem

ORDERED LIST <h6>

1. font: Source Sans Pro
2. font-size: 1.7rem
3. line-height: 1.5rem

HTML CODE with class attributes

|  |
| --- |
| <div class="usa-grid-full">  <div class="usa-width-one-third">  <h6 class="usa-heading-alt">Unordered list</h6>  <ul>  <li>Unordered list item</li>  <li>Unordered list item</li>  <li>Unordered list item</li>  </ul>  </div>  <div class="usa-width-one-third">  <h6 class="usa-heading-alt mt0">Ordered list</h6>  <ol>  <li>Ordered list item</li>  <li>Ordered list item</li>  <li>Ordered list item</li>  </ol>  </div>  </div> |

## Color Palette

The MCCF TASCore color palette is based on palettes used by USWDS and the VA Playbook. These colors support the highest Section 508 color contrast requirements while also promoting trust by invoking a professional look and feel. It is a simple set of colors that use shades of cool blues and grays-combined with definitive styling to allow users to feel welcomed.

The **primary colors** are shades of blue and gray that are the dominant colors of the site. Also, the use of white/white space should be used so content will be evenly spaced. This will promote sincerity, calmness and professionalism.

**Secondary colors** are used to promote important highlighted features on a page such as buttons and list headers.

Shades of gray are used as **background colors** for larger content areas. Be sure to use enough contrast between adjacent colors.

Lastly, the **tertiary colors** are available for content driven messages and alerts. Use sparingly and do not overpower the primary colors. See color chips below with associated hex values.

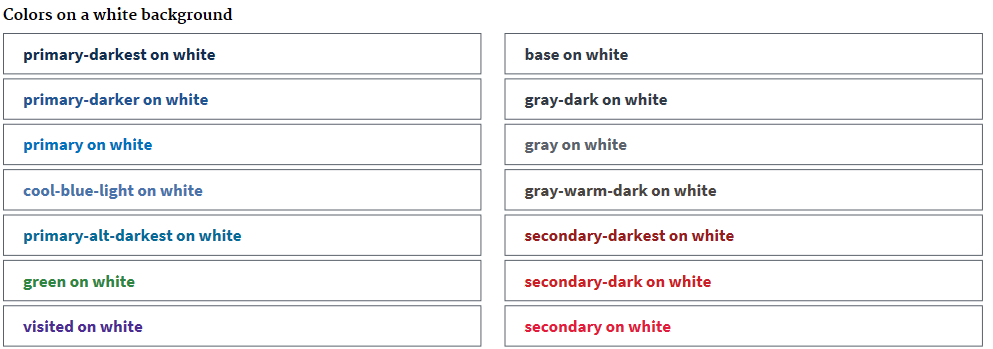
|  |  |
| --- | --- |
| **Primary colors** | **Secondary Colors** |
| |  |  |  | | --- | --- | --- | | **#0071bc** | **#205493** | **#112e51** | | **#212121** | **#323a45** | **#aeb0b5** | | |  |  |  |  |  | | --- | --- | --- | --- | --- | | **#02bfe7** | **#046b99** | **#00A6D2** | **#9bdaf1** | **#e1f3f8** | | **#e31c3d** | **#981b1e** | **#cd2026** | **#e59393** | **#f9dede** | |
| **Tertiary Colors** | **Background colors** |
| |  |  |  | | --- | --- | --- | | **#fdb81e** | **#2e8540** | **#205493** | | **#f9c642** | **#4aa564** | **#4773aa** | | **#e7f4e4** | **#fff1d2** | **#e31c3d** | | |  |  |  |  |  | | --- | --- | --- | --- | --- | | **#32a45** | **#5b616b** | **#aeb0b5** | **#d6d7d9** | **#f1f1f1** | | **#494440** | **#e4e2e0** | **#ffffff** | **#dce4ef** | **#fdb81e** Breadcrumb | | **#003e73** Secondary footer | **#112e51**  Primary footer |  |  |  | |

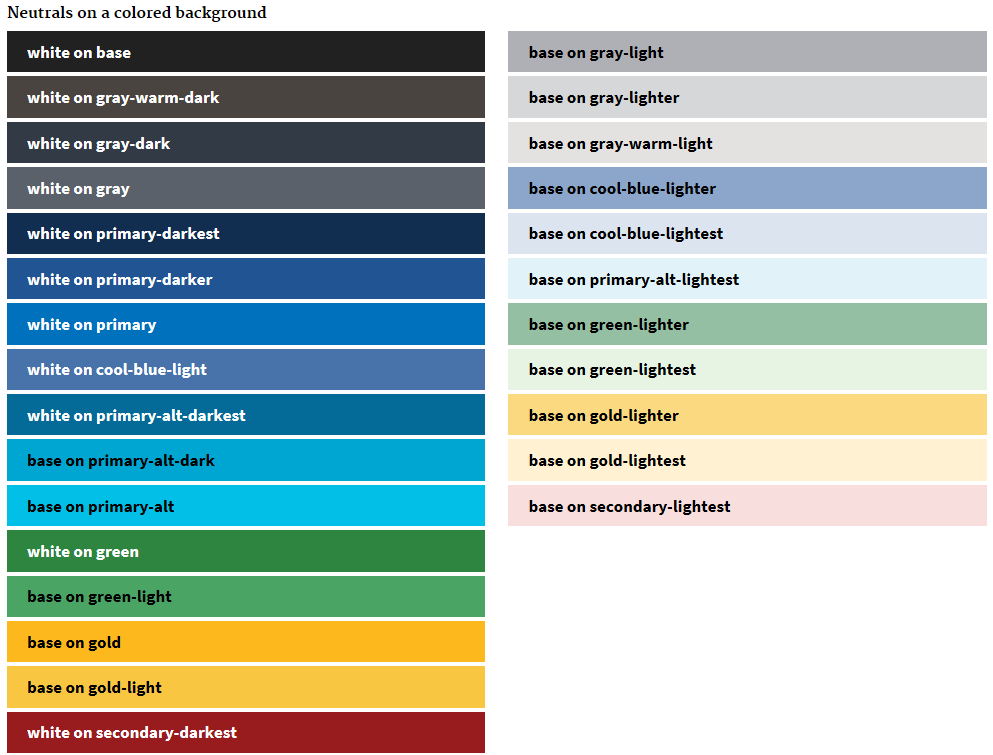
## Text Accessibility

By following the styles of Web Content Accessibility Guidelines (WCAG), content on web pages will accessible by everyone regardless of the disability. This set of standards provides a contrast ratio of 4.5 to 1 so text and colors can be readable even by those with color blindness.

To confirm contrast ratio is Section 508 compliant, this [link](https://webaim.org/resources/contrastchecker/) will confirm if the contrast ratio is high enough between a text color and background color. The [WebAIM](https://webaim.org/resources/contrastchecker/) site has online tool for testing ratios between the two colors. If the ratio is 4.5 or higher, then the colors are Section 508 compliant.

Below is a chart of color combinations that can be used for text color with background colors. These are all part of USWDS style guides and are all compliant with Section 508 requirements.





## Headers

Headers are used to display logos, branding and a horizontal navigation bar. It’s primary purpose is allow users to where they are and to navigate across different sections of a website.

## Accessibility

Developers need to follow certain rules when creating headers to maintain Section 508 compliance. Below is a list of accessibility rules that must be followed to achieve compliance:

Include skip navigation links so screen readers can bypass long navigation lists

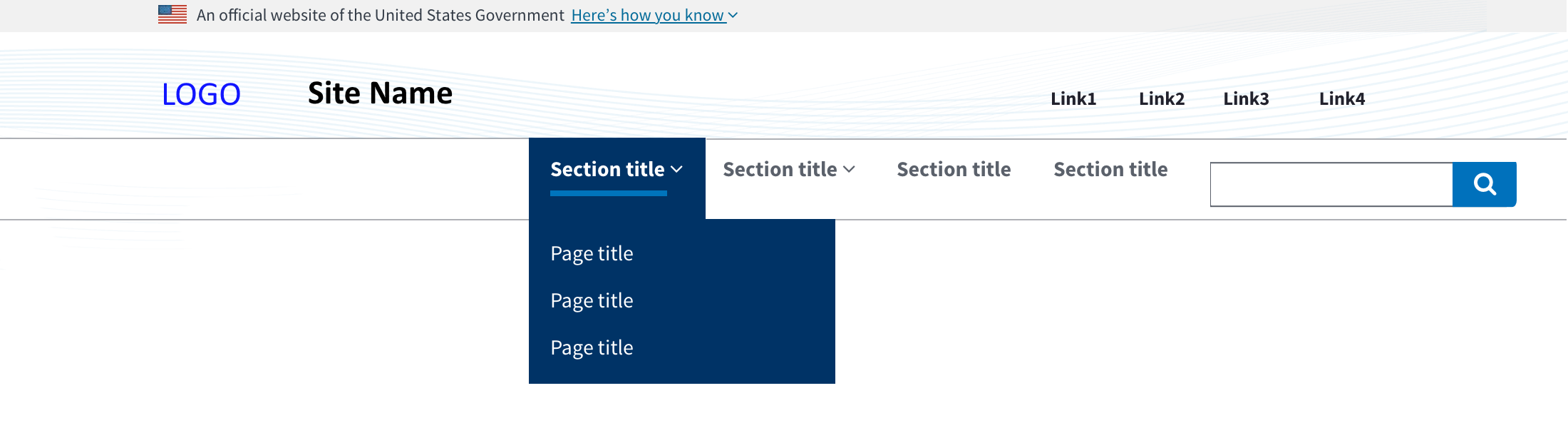
* Include Tab focus for all top-level navigation
* Ensure horizontal navigation is keyboard compatible; test using the Tab key
* Avoid using hover to expand dropdown lists, this does not work on touch screens. Drop downs should expand on click or with keyboard navigation
* Use list for navigation links- this helps screen readers better decipher header contents
* Include alt tags for logos in header
* If the logo is text, use em as opposed to <h1> markup. In CSS, em is relative to the font-size of the element (2em means 2 times the size of the current font)

## Usability

* All headers should include “Official Government Site” banner and logo and site name. The banner helps to identify the site as a federal while the logo helps visitors understand the organization.



* A horizontal navigation bar is the most useful control to help users navigate through the site.
* Use https protocol for all federal websites that will support security, privacy, and encryption.
* List all website sections as links in the horizontal navigation bar.
* Use dropdown menus for larger size sites.
* Use descriptive, recognizable link labels and keep the text clear and uncluttered.
* Left justify all links.
* Present links in priority order with higher demand links on the left and lesser ones towards the right within the navigation bar.
* Show users where they are within the menu options by highlighting the current selection.
* Review business requirements within your group to better understand the information architecture.

Image of extended header with background image, navigation bar, search input and quick links.

## Header and TAS custom styling

|  |  |
| --- | --- |
|  | Tas-custom.css |
| Quick Links | .usa-nav-secondary-links a:hover, .usa-nav-secondary-links .usa-header-search-button:hover {  color: #000!important;  text-decoration: underline; } |
| Logo | .topVALogo {  float:left;  padding-right:5px;  } |
| WebTitle | .usa-logo-text h1 {  font-size: 30px;  clear:none !important;  margin-top: 1.0em !important; } |
|  | /\* background image for header \*/  .usa-header {  border-bottom: 1px solid #d6d7d9;  background-image:url("/assets/img/thread4.png");  background-position:left;  background-repeat: repeat-x; } |

## Footers

Footers are always located on the bottom of every web page that contain a group of internal and external hyperlinks. It also promotes branding, can include logos of smaller size which are generally located on the bottom right side.

## Accessibility

* Users must be allowed to navigate every link in the footer using the tab key.
* On smaller screens, footers should collapse so links are still readable and do not overlap.

## Usability

* It’s a good idea to include the main global navigation links in the footer.
  + Group footer links when possible into columns.
* Logos should be smaller and located on either the bottom left or right of the page.
* Contact information should point to a general email address; never use a personal email address.
* Use social media links that point only to your agency.
* Include a “Return to Top” hyperlink above the main footer content.
* Top footer uses background color is #003e73.



Generic footer with primary and secondary links, logo and site name.

|  |  |
| --- | --- |
| .usa-footer-secondary\_section {  background-color: #003e73 !important;  } | TAS-custom.css |

## Side Navigation

Side navigations are vertical links that are used to navigate to subsection of the same website.

|  |  |
| --- | --- |
|  | <h6 class="usa-heading-alt">Single level</h6>  <div class="usa-grid-full">  <aside class="usa-width-one-fourth">  <ul class="usa-sidenav-list">  <li>  <a class="usa-current" href="javascript:void(0);">Current page</a>  </li>  <li>  <a href="javascript:void(0);">Parent link</a>  </li>  <li>  <a href="javascript:void(0);">Parent link</a>  </li>  </ul>  </aside>  </div> |
|  | <h6 class="usa-heading-alt">Two levels</h6>  <div class="usa-grid-full">  <aside class="usa-width-one-fourth">  <ul class="usa-sidenav-list">  <li> <a href="javascript:void(0);">Parent link</a></li>  <li>  <a class="usa-current" href="javascript:void(0);">Current page</a>  <ul class="usa-sidenav-sub\_list">  <li><a href="javascript:void(0);">Child link</a> </li>  <li><a href="javascript:void(0);">Child link</a> </li>  <li><a href="javascript:void(0);">Child link</a> </li>  <li><a href="javascript:void(0);">Child link</a> </li>  <li><a class="usa-current" href="javascript:void(0);">Child Lin </li>  </ul>  </li>  <li><a href="javascript:void(0);">Parent link</a></li>  </ul>  </aside>  </div> |
|  | <h6 class="usa-heading-alt">Three levels</h6>  <div class="usa-grid-full">  <aside class="usa-width-one-fourth">  <ul class="usa-sidenav-list">  <li><a href="javascript:void(0);">Parent link</a> </li>  <li>  <a class="usa-current" href="javascript:void(0);">Current page</a>  <ul class="usa-sidenav-sub\_list">  <li><a href="javascript:void(0);">Child link</a></li>  <li>  <a href="javascript:void(0);">Child link</a>  <ul class="usa-sidenav-sub\_list">  <li><a href="javascript:void(0);">Grandchild link</a></li>  <li><a href="javascript:void(0);">Grandchild link</a> </li>  <li><a class="usa-current" href="javascript:void(0);">Grandchild link</a></li>  </ul>  </li>  <li><a href="javascript:void(0);">Child link</a></li>  <li><a href="javascript:void(0);">Child link</a></li>  <li><a href="javascript:void(0);">Child link</a></li>  </ul>  </li>  <li><a href="javascript:void(0);">Parent link</a></li>  </ul>  </aside>  </div> |

## Accessibility

* Each link in the side navigation must be accessible using the Tab key.

## Usability

* Must be displayed on the left side of web page under the header and above the footer.
* Include links to subsections of the website.
* Keep navigation links simple and easy to read; do not clutter with too much text.
* Side navigation should not go deeper than three levels otherwise users will get lost.

## Expand/Collapse Side Navigation

* Expand/Collapsible side navigation is another option for side navigation links. This helps structure content using menus links with indented submenu links.
* Menu links can be dynamically rendered on page load using JSON data.
* Tree view navigation is a shared component that is fully accessible. Each link is fully accessible using the tab key

|  |  |
| --- | --- |
|  | Font-family: Source Sans Pro  Line-height:26px  Underline link on mouse hover and change color to black  Arrow-right: #000  Arrow-down: #fdb81e |

## Grids

Grids are used for responsive web design. TASCore uses a 12-column grid CSS structure so the web page can be viewable regarding of the viewport size. Web components will collapse in a stackable format as the page size is reduced or the screen size is smaller.

Grids are applied to div elements using the CSS usa-grid class.

Grids without any padding on left or right use usa-grid-full class.

Grid class are semantic in name by its width.

Grids are at full-width with smaller screens.

usa-width-one-half = ½ web page

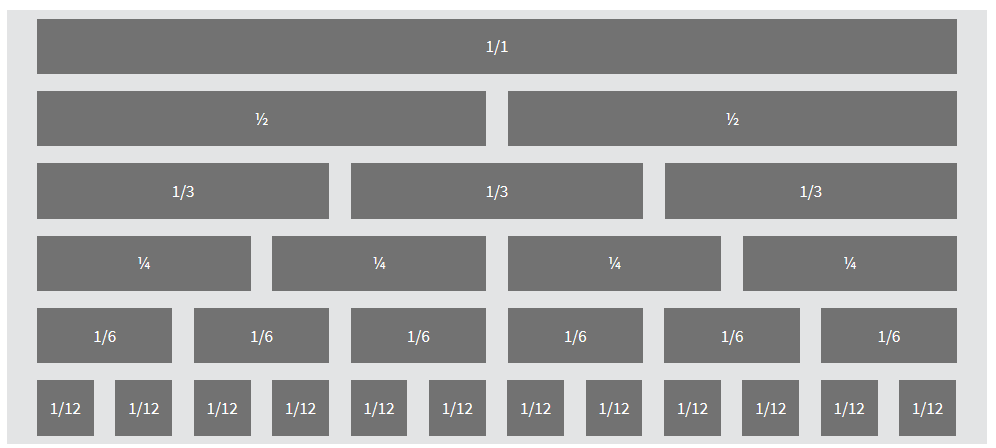
usa-width-one-third = 1/3 web page

usa-width-one-fourth = ¼ web page

usa-width-two-thirds = 2/3 web page

usa-width-one-sixth = 1/6 web page

usa-width-one-twelfth = 1/12 web page



## Accessibility

* Users should be able to increase font size by 200% without breaking the layout.

## Usability

* The entire web site must use the responsive grid system for the header, main body, global navigation, side navigation and footer.
* Use the 12-column grid with flexible columns widths; padding is dependent on screen sizes.
* Avoid using text longer than **75 characters**. Put text in a smaller grid column to avoid long lines of text running across the page.

## Buttons

Buttons are a call to action that provides functionality to web page. Buttons are categorized into two groups: **primary and secondary**.

Use the usa-button class for buttons, with the exception of search (see section 3.21 Search Bar).

|  |  |  |
| --- | --- | --- |
| usa-button |  | <button class="usa-button">Default</button> |
| usa-button usa-button-hover |  | <button class="usa-button usa-button-hover">Hover</button> |
| usa-button usa-button-active |  | <button class="usa-button usa-button-active">Active</button> |
| usa-button usa-focus |  | <button class="usa-button usa-focus">Focus</button> |
| usa-button |  | <button class="usa-button" disabled="">Disabled</button> |
| usa-button-secondary |  | <button class="usa-button-secondary">Default</button> |
| usa-button-secondary usa-button-hover |  | <button class="usa-button-secondary usa-button-hover">Hover</button> |
| usa-button-secondary usa-button-active |  | <button class="usa-button-secondary usa-button-active">Active</button> |
| usa-button-secondary usa-focus |  | <button class="usa-button-secondary usa-focus">Focus</button> |
| usa-button-secondary |  | <button class="usa-button-secondary" disabled="">Disabled</button> |
| usa-button-primary-alt |  | <button class="usa-button-primary-alt">Default</button> |
| usa-button-gray |  | <button class="usa-button-gray">Default</button> |
| usa-button-big |  | <button class="usa-button-big">Default</button> |

## Accessibility

* When focused on Tab, button should display visible focus state.
* Do **not** use <div> or <img> tags to create buttons – they cannot be read by screen-readers.
* Pressing space bar triggers a button; clicking the enter key triggers a hyperlink.

## Usability

Buttons should be used for important interactivity on the site such as “Download”, “Submit” or “Log Out”

Do not use buttons to navigate web pages; use hyperlinks

## Labels

Labels draw attention to new or important content. 

## Accessibility

Use ARIA live regions to alert screen readers of the change.

## Usability

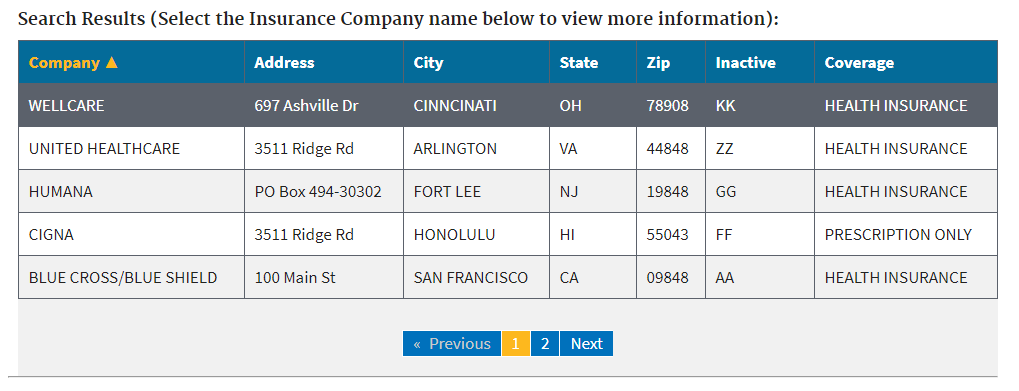
When to use:

* To draw attention to new or important content
* To filter results with one or more tags
* To indicate the number of new or unread items within a container. Like a badge, to display the number of unread emails as an example.
* Labels are non-interactive, there is no hover, on focus or click function.
* Use sparingly.

|  |  |
| --- | --- |
| **Code** | <span class="usa-label ">New</span>  <span class="usa-label ">Important</span> <span class="usa-label ">4</span> |

## Data Tables

Tables are used to show tabular data in columns and rows. It is a display of information in tabular form with rows and columns. Data table are populated from a database. TAS Core tables contain bordered styles. The header columns come with a SORT feature that can toggle between ascending and descending the data in the column. The table also has a paginator UI control – see section 3.29.



|  |
| --- |
| **SAMPLE CODE for data table with sort and pagination** |
| <table class="table table-hover table\_blur" id="dataTable" tabindex="0">  <thead>  <tr role="row">  <th scope="col" id="dt-header-Insurance-Company"> Insurance Company<span>▼</span></th>  <th scope="col" id="dt-header-Address">Address</th>  <th scope="col" id="dt-header-City">City</th>  <th scope="col" id="dt-header-State">State</th>  <th scope="col" id="dt-header-ZIP">ZIP</th>  <th scope="col" id="dt-header-Inactive">Inactive</th>  <th scope="col" id="dt-header-Type-of-Coverage">Type of Coverage</th>  </tr>  </thead>  <tbody>  <tr tabindex="0" id="dataTableRow0" class="active">  <td headers="dt-header-Insurance-Company">AETNA</td>  <td headers="dt-header-Address">3511 RIDGE RD</td>  <td headers="dt-header-City">SAN FRANCISCO</td>  <td headers="dt-header-State">CA</td>  <td headers="dt-header-ZIP">90120</td>  <td headers="dt-header-Inactive">IEN</td>  <td headers="dt-header-Type-of-Coverage">PRESCRIPTION ONLY</td>  </tr>  <tr tabindex="0" id="dataTableRow1">  <td headers="dt-header-Insurance-Company">AETNA</td>  <td headers="dt-header-Address">7 LITTLE BROOK LN</td>  <td headers="dt-header-City">ARLINGTON</td>  <td headers="dt-header-State">VA</td>  <td headers="dt-header-ZIP">44848</td>  <td headers="dt-header-Inactive">IEN</td>  <td headers="dt-header-Type-of-Coverage">HEALTH INSURANCE</td>  </tr>  <tr tabindex="0" id="dataTableRow2" class="">  <td headers="dt-header-Insurance-Company">BLUE CROSS/BLUE SHIELD</td>  <td headers="dt-header-Address">100 BROADWAY</td>  <td headers="dt-header-City">NEW YORK City</td>  <td headers="dt-header-State">NY</td>  <td headers="dt-header-ZIP">10019</td>  <td headers="dt-header-Inactive">IEN</td>  <td headers="dt-header-Type-of-Coverage">HEALTH INSURANCE</td>  </tr>  <tr tabindex="0" id="dataTableRow3">  <td headers="dt-header-Insurance-Company">BLUE CROSS/BLUE SHIELD/FL</td>  <td headers="dt-header-Address">PO BOX 494-30302</td>  <td headers="dt-header-City">HONOLULU</td>  <td headers="dt-header-State">HI</td>  <td headers="dt-header-ZIP">55043</td>  <td headers="dt-header-Inactive">IEN</td>  <td headers="dt-header-Type-of-Coverage">PRESCRIPTION ONLY</td>  </tr>  <tr tabindex="0" id="dataTableRow4">  <td headers="dt-header-Insurance-Company">FLORIDA BLUE</td>  <td headers="dt-header-Address">1600 PARKER AVE</td>  <td headers="dt-header-City">FORT LEE</td>  <td headers="dt-header-State">NJ</td>  <td headers="dt-header-ZIP">09848</td>  <td headers="dt-header-Inactive">IEN</td>  <td headers="dt-header-Type-of-Coverage">HEALTH INSURANCE</td>  </tr>  </tbody>  </table> |

## Accessibility

* Tables must have two levels of headers. Each header cell should have scope=”col” or scope=”row”
* Complex tables should have a unique id and each data cell should have a headers attribute with the header’s cell id listed.
* Use the <caption> tag inside the <table> element when adding titles.

## Usability

* Use tables to display tabular data.
* Also, consider using lists depending on the content.
* Do not add too many columns – no horizontal scroll bar should appear on browser window.

## Alerts

Alerts keep users informed of important and time sensitive changes.

There are 4 types of alerts, each of them color coded accordingly to the alert type:

Success - green

Warning – yellow

Error – red

Information – blue

|  |  |
| --- | --- |
|  |  |

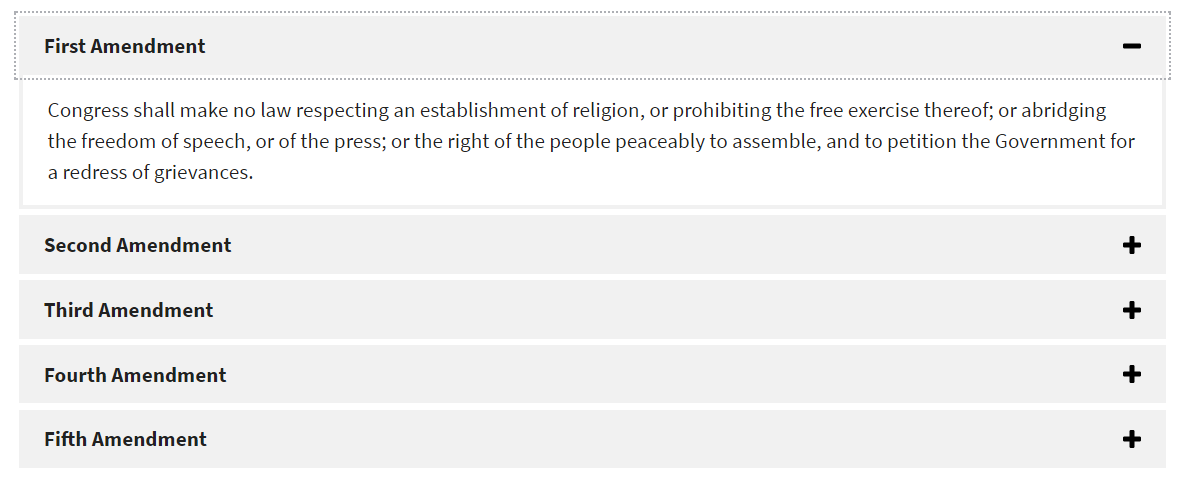
TASCore alerts use [PrimeNG](https://www.primefaces.org/primeng/#/growl) component called Growl, which is part of the Angular suite of components.

The Growl messages needed to be restyled to meet the USWDS so custom classes were created inside the tas-custom.css file.

|  |  |
| --- | --- |
|  | .ui-growl-item-container.ui-state-highlight.ui-growl-message-success {  z-index:5000 !important;  background-color: #e7f4e4;  font-family: "Merriweather","Times New Roman";  border-left: 8px solid #2e8540;  } |
|  | .ui-growl-item-container.ui-state-highlight.ui-growl-message-warn {  background-color: #fff1d2;  z-index:5000 important;  font-family: "Merriweather","Times New Roman"; border-left: 8px solid #fdb81e;  } |
|  | .warning-link {  position: fixed !important;  margin-top: -100px !important;  z-index: 999999;  left:20px !important; } |
|  | .ui-growl-item-container.ui-state-highlight.ui-growl-message-info {  background-color: #e1f3f8;  z-index:5000 important;  border-left: 8px solid #02bfe7;  font-family: "Merriweather", "Times New Roman";  } |
|  | .ui-growl-item-container.ui-state-highlight.ui-growl-message-error {  background-color: #f9dede;  z-index:5000 important;  border-left: 8px solid #e31c3d;  font-family: "Merriweather","Times New Roman";  } |
| Message font | .ui-growl-message-success p, .ui-growl-message-warn p, .ui-growl-message-error p, .ui-growl-message-info p{  font-family: "Source Sans Pro";  } |

## Accordions

Accordions are web controls that can expand and collapse. They provide a way of making content less heavy. Made up of multiple items containing a head/title and a body with expanded content, the accordion is an interactive component that is also responsive.



## Accessibility

* Accordion header should be coded as <buttons> so they are read by screen readers and used by the keyboard.
* If accordion item is expanded, the button should be stated with a aria-expanded=”true”. Other items in the accordion will have aria-expanded=”false” attribute when the accordion is initialized on the page load.
* Each button has unique name aria-controls=”id” that associates the control to the region by referencing the element’s id.
* Each content area will have its aria-hidden attribute set to true or false depending on the button’s aria-expanded attribute. To ensure accessibility, do not set aria-hidden=”true”.

## Usability

* Information needs to be displayed in a small space.
* Use well-formatted text.
* Users only need a few specific pieces of information.
* Allow users to click anywhere in the header to expand or collapse the content.

|  |
| --- |
| <h3>Accordion</h3>  <div style="margin-bottom: 1em">  <button type="button" aria-label="Previous" title="Previous" class="fa fa-chevron-up" (click)="openPrev()"></button>  <button type="button" aria-label="Next" title="Next" href="#" class="fa fa-chevron-down" (click)="openNext()"></button>  </div>  <div role="tablist">  <p-accordion >  <p-accordionTab header="Godfather I" role="tab" aria-expanded="true" aria-controls="a1">  <div id="a1" aria-hidden="false">  Content goes here.  </div>  </p-accordionTab>  <p-accordionTab header="Godfather II" role="tab" aria-expanded="false" aria-controls="a2">  <div id="a2" aria-hidden="false">  Content goes here.  </div>  </p-accordionTab>  <p-accordionTab header="Godfather III" role="tab" aria-expanded="false" aria-controls="a3">  <div id="a3" aria-hidden="false"> Content goes here.  </div>  </p-accordionTab>  <p-accordionTab header="Godfather IV" role="tab" aria-expanded="false" aria-controls="a4">  <div id="a4" aria-hidden="false"> More content goes here.  </div>  </p-accordionTab>  <p-accordionTab header="Godfather V" role="tab" aria-expanded="false" aria-controls="a5">  <div id="a5" aria-hidden="false"> Even more content goes here.  </div>  </p-accordionTab> |

## Form Controls

Allows users to enter information on a web page using text input fields, drop downs, checkboxes, radio buttons, date inputs and validation alerts.

## Accessibility

* Display form controls in same order in HTML as they appear on screen. Screen readers narrate forms in the order they appear in HTML.
* Align validation messages with input fields so people using magnifiers can read them quickly.
* Group related controls in a fieldset element. Use the legend element to offer a label within each one. This makes it easier for screen reader users to navigate the form.
* Use a single legend for a fieldset as required. A good example would be a question with radio button options for answers. The question and radio buttons are wrapped in a fieldset while the question itself is inside the legend tag.
* Keep form layout in a vertical layout – this makes it easier to read.

## Text Input

Text input allows people to enter any combination of alpha-numeric characters. Inputs can be either single line or multi lines.

|  |  |
| --- | --- |
|  | <label for="input-type-text">Text input label</label>  <input id="input-type-text" name="input-type-text" type="text">  <label for="input-focus">Text input focused</label>  <input class="usa-focus" id="input-focus" name="input-focus" type="text">  <div class="usa-input-error">  <label class="usa-input-error-label" for="input-error">Text input error</label>  <span class="usa-input-error-message" id="input-error-message" role="alert">Helpful error message</span>  <input id="input-error" name="input-error" type="text" aria-describedby="input-error-message">  </div>  <label for="input-success">Text input success</label>  <input class="usa-input-success" id="input-success" name="input-success" type="text">  <label for="input-type-textarea">Text area label</label>  <textarea id="input-type-textarea" name="input-type-textarea"></textarea> |

## Accessibility

* Avoid using placeholder text. This text does not provide high enough contrast ratio.
* Avoid breaking numbers using separate fields such as SS#, phone or credit card numbers. Use one input field for phone number; not three. Each field needs to be labeled for a screen reader.

## Usability

* Allow users to paste in a response.
* Use when it’s easiest to input data – for example use text fields over date fields when entering a birthdate.
* Use when you can’t reasonably predict a user’s response to an answer.
* Use single line text fields for shorter answers; use multiline fields for answers that require paragraphs of text.
* Show validation messages after a user has entered data in a field.

## Dropdown

A drop down allows a user to select one or more options from a list.

|  |  |
| --- | --- |
|  | <form class="usa-form">  <label for="options">Dropdown label</label>  <select name="options" id="options">  <option value>- Select -</option>  <option value="value1">Option A</option>  <option value="value2">Option B</option>  <option value="value3">Option C</option>  </select>  </form> |

## Accessibility

* Dropdown must include a label. Use something like “Select a xxxx” as the default menu option.
* Don’t use JavaScript to automatically submit the form when an option is selected. This disrupts screen readers because they select each option as they are read.

## Usability

* Use sparingly – no more than 15 possible options if space is limited.
* If the list of options is short use radio buttons instead.
* If the list of options is very long, use input text fields instead.
* Be sure to test the dropdown – this control is used as “UI of the last resort” because many users find them confusing and difficult.
* Use a submit button at the end of the form – do not use JavaScript to automatically submit the form.

## Checkboxes

Checkboxes allow users to select one or more options from a visible list.

|  |  |
| --- | --- |
|  | <fieldset class="usa-fieldset-inputs usa-sans">  <legend class="usa-sr-only">Historical figures 1</legend>  <ul class="usa-unstyled-list">  <li>  <input id="truth" type="checkbox" name="historical-figures-1" value="truth" checked>  <label for="truth">Sojourner Truth</label>  </li>  <li>  <input id="douglass" type="checkbox" name="historical-figures-1" value="douglass">  <label for="douglass">Frederick Douglass</label>  </li>  <li>  <input id="washington" type="checkbox" name="historical-figures-1" value="washington">  <label for="washington">Booker T. Washington</label>  </li>  <li>  <input id="carver" type="checkbox" name="historical-figures-1" disabled>  <label for="carver">George Washington Carver</label>  </li>  </ul>  </fieldset> |

## Accessibility

* Surround the set of checkboxes with a <fieldset>. Use a <legend> element to provide context for the grouping.
* Custom checkboxes are accessible to screen readers because default checkboxes are moved off-screen with position: absolute; left: -999em.
* Each input should have an Id attribute while the corresponding label should be same value in the for attribute.
* The title attribute can replace <label>.

## Usability

* When a user can select from a short list of choices.
* “Yes” or “No” toggle when only one option is available. Used to toggle a setting on or off.
* When users need to see all available options at a glance.
* If a user can only select one option, use radio buttons instead.
* Users should be able to select the button or text label to select or deselect an option.
* List options vertically for easier screen readability.
* Avoid using negative language.
* Make sure selections are spaced enough for touch screens.

## Radio buttons

Radio buttons allow users

|  |  |
| --- | --- |
|  | <fieldset class="usa-fieldset-inputs usa-sans">  <legend class="usa-sr-only">Historical figures 2</legend>  <ul class="usa-unstyled-list">  <li>  <input id="stanton" type="radio" checked name="historical-figures-2" value="stanton">  <label for="stanton">Elizabeth Cady Stanton</label>  </li>  <li>  <input id="anthony" type="radio" name="historical-figures-2" value="anthony">  <label for="anthony">Susan B. Anthony</label>  </li>  <li>  <input id="tubman" type="radio" name="historical-figures-2" value="tubman">  <label for="tubman">Harriet Tubman</label>  </li>  </ul>  </fieldset> |

to select from a list of available choices and only select one option.

## Accessibility

* Group radio buttons inside a <fieldset> and the describe the group with a <legend>.
* Each radio button must have a <label>. The <for> attribute should match input <id>.
* Title attribute can replace <label>.

## Usability

* Use when only one option can be selected from the available choices.
* Use when number of options can fit on mobile device.
* Do not use if users need to select more than one option.
* Users should be able to click label text or the button itself to select or deselect.
* Try to not preset a default value – this leads to confusion.

## Date input

Use three input fields for date. Month Day Year.

|  |  |
| --- | --- |
|  | <fieldset>  <legend>Date of birth</legend>  <span class="usa-form-hint" id="dobHint">For example: 04 28 1986</span>  <div class="usa-date-of-birth">  <div class="usa-form-group usa-form-group-month">  <label for="date\_of\_birth\_1">Month</label>  <input class="usa-input-inline" aria-describedby="dobHint" id="date\_of\_birth\_1" name="date\_of\_birth\_1" type="number" min="1" max="12" value="">  </div>  <div class="usa-form-group usa-form-group-day">  <label for="date\_of\_birth\_2">Day</label>  <input class="usa-input-inline" aria-describedby="dobHint" id="date\_of\_birth\_2" name="date\_of\_birth\_2" type="number" min="1" max="31" value="">  </div>  <div class="usa-form-group usa-form-group-year">  <label for="date\_of\_birth\_3">Year</label>  <input class="usa-input-inline" aria-describedby="dobHint" id="date\_of\_birth\_3" name="date\_of\_birth\_3" type="number" min="1900" max="2000" value="">  </div>  </div>  </fieldset> |

## Accessibility

* Text fields should follow the accessibility guidelines to input fields.
* Do not use JavaScript to advance to next field; this makes it difficult for keyboard navigators to correct mistakes.

## Usability

* Use this format to enter dates.
* Make sure fields are properly labeled

## Validation

Validation is used to give users a notification message on how data should be entered inside input fields.

|  |  |
| --- | --- |
|  | <form class="usa-form">  <fieldset>  <legend class="usa-drop\_text">Enter a code</legend>  <div class="usa-alert usa-alert-info">  <div class="usa-alert-body">  <h3 class="usa-alert-heading">Codes must:</h3>  </div>  <ul class="usa-checklist" id="validate-code">  <li data-validator="uppercase">Have at least 1 uppercase character</li>  <li data-validator="numerical">Have at least 1 numerical character</li>  </ul>  </div>  <label for="code">Code</label>  <input id="code" name="code" type="text"  aria-describedby="validate-code"  data-validate-uppercase="[A-Z]"  data-validate-numerical="\d"  data-validation-element="#validate-code">  <input type="submit" value="Submit code">  </fieldset>  </form> |

## Accessibility

* Follow all the same 508 compliance rules as regular input fields.

## Usability

* Input fields with custom validation logic provide helpful feedback to users if assigned a data-validation-element attribute set to a CSS selector:

data-validation-element="#validate-code"

* Do not use spaces in the name of the validator: uppercase
* Set the data-validator attribute to the name of the validator
  + data-validator=”uppercase”
* On input fields, add a field called data-validate-validator name and set its value to an expression to represent the validator’s condition.
  + data-validate-uppercase=”[A-Z]”

## Form Templates

Forms are a group of web controls used to enter data.

## Accessibility

* All form controls must meet the accessibility guidelines for each control.
* Display form controls in the same order in the HTML as they appear on screen. This helps screen readers narrate form controls in the order they appear in HTML.
* Align validation message with input fields so people with magnifiers can read them quickly.
* Group controls inside a fieldset element. Fieldset and legend elements make it easier for screen reader user to navigate the form.
* Use a single legend for fieldset – this is required.
* Embed multiple fieldsets and legends for more complex forms.
* Keep form blocks in vertical pattern.

|  |  |
| --- | --- |
|  | <form class="usa-form">  <fieldset>  <legend>Name</legend>  <label for="title" class="usa-input-optional">Title</label>  <input class="usa-input-tiny" id="title" name="title" type="text">  <label for="first-name">First name</label>  <input id="first-name" name="first-name" type="text" required="" aria-required="true">  <label for="middle-name" class="usa-input-optional">Middle name</label>  <input id="middle-name" name="middle-name" type="text">  <label for="last-name">Last name</label>  <input id="last-name" name="last-name" type="text" required="" aria-required="true">  </fieldset>  </form> |

* Each field must follow the accessibility guidelines for all form controls.
* Use this form when collecting the user’s full name.
* Do not restrict the types of characters users can enter in the fields.

|  |  |
| --- | --- |
|  | <form class="usa-form-large">  <fieldset>  <legend>Mailing address</legend>  <label for="mailing-address-1">Street address 1</label>  <input id="mailing-address-1" name="mailing-address-1" type="text">  <label for="mailing-address-2" class="usa-input-optional">  Street address 2</label>  <input id="mailing-address-2" name="mailing-address-2" type="text">  <div>  <div class="usa-input-grid usa-input-grid-medium">  <label for="city">City</label>  <input id="city" name="city" type="text">  </div>  <div class="usa-input-grid usa-input-grid-small">  <label for="state">State</label>  <select id="state" name="state">  <option value>- Select -</option>  <option value="AL">Alabama</option>  <option value="AK">Alaska</option>  <option value="AZ">Arizona</option>  <option value="AR">Arkansas</option>  …  <option value="VA">Virginia</option>  <option value="WA">Washington</option>  <option value="WV">West Virginia</option>  <option value="WI">Wisconsin</option>  <option value="WY">Wyoming</option>  <option value="AA">AA - Armed Forces Americas</option>  <option value="AE">AE - Armed Forces Africa</option>  <option value="AE">AE - Armed Forces Canada</option>  <option value="AE">AE - Armed Forces Europe</option>  <option value="AE">AE - Armed Forces Middle East</option>  <option value="AP">AP - Armed Forces Pacific</option>  </select>  </div>  </div>  <label for="zip">ZIP</label>  <input class="usa-input-medium" id="zip" name="zip" type="text" pattern="[\d]{5}(-[\d]{4})?">  </fieldset></form> |

* Each field must follow the accessibility guidelines for all form controls.
* For the ZIP field, make sure to include pattern attribute so a hyphen is entered before the four digit extension.
  + <label for="zip">ZIP</label>

<input class="usa-input-medium" id="zip" name="zip" type="text" pattern="[\d]{5}(-[\d]{4})

* Label the optional inputs.
* Let users type in state abbreviation on the drop-down menu.
* Support both 5 and 9-digit ZIP codes.

Use [Politspace](https://www.filamentgroup.com/lab/politespace.html) as a way to implement input masking zip code input fields.

* Each field must follow the accessibility guidelines for all form controls.
* Do not sign out users without giving them 20 second’s notice to request more time.
* Use this form when authentication and authorization are required to access private data.
* Less is more – keep it simple and use less intrusive text.
* Allow users to use their email address to sign in.
* Allow for a “Remember Me” check box on trusted computers so it’s easier to sign in next time.
* Make it easy for users to retrieve a forgotten username or password.
* Allow users to see their passwords using a hyperlink to display password. Avoid complete password masking.

|  |  |
| --- | --- |
|  | <form class="usa-form">  <fieldset>  <legend class="usa-drop\_text">Reset password</legend>  <span class="usa-serif">Please enter your new password</span>  <div class="usa-alert usa-alert-info">  <div class="usa-alert-body">  <h3 class="usa-alert-heading">Password information</h3>  <p class="usa-alert-text">  Length requirements  <br>  Character constraints, if any  </p>  </div>  </div>  <label for="password-reset">New password</label>  <input id="password-reset" name="password" type="password">  <label for="confirmPassword">Confirm password</label>  <input id="confirmPassword" name="confirmPassword" type="password">  <p class="usa-form-note">  <a href="javascript:void(0);"  class="usa-show\_multipassword"  aria-controls="password-reset confirmPassword">Show my typing</a>  </p>  <input type="submit" value="Reset password">  </fieldset>  </form> |

* Each field must follow the accessibility guidelines for all form controls.
* Offer a way to easily reset a password if a user forgets the username or password.
* State any password requirements up front.
* See [NIST’s strength of memorized secrets](https://github.com/usnistgov/800-63-3/blob/nist-pages/sp800-63b/appA_memorized.md)**.**

## Search Bar

An input field that allows users to search for specific content using keywords and/or phrases or if content cannot be found in the main navigation. TAS currently used 2 different search input fields.

* Site-wide search
* Contact search

The search field on the TAS application is located in the header and can also be found on the Contacts widget both located on the TAS home page.

|  |  |
| --- | --- |
| **SITE WIDE SEARCH** |  |
|  | <div role="search">  <label class="usa-sr-only" for="search-field-smallsite-wide-search">site-wide-search</label>  <input placeholder="Enter Search"  type="search" id="search-field-smallsite-wide-search"  aria-label="site-wide-search"  class="ng-untouched ng-pristine ng-valid"  title="Enter search term.">  <button type="submit" disabled="">  <span class="usa-sr-only">Search</span>  </button>  </div> |
| **CONTACT SEARCH** |  |
|  | <div role="search">  <label class="usa-sr-only" for="search-field-smallcontact-search">contact-search</label>  <input placeholder="Enter Search"  type="search" id="search-field-smallcontact-search"  aria-label="contact-search"  class="ng-untouched ng-pristine ng-valid"  title="Enter search term.">  <button type="submit" disabled="">  <span class="usa-sr-only">Search</span>  </button> |

## Accessibility

* Each field must follow the accessibility guidelines for all form controls.
* Always include the word “search” inside the <button> element for screen readers. The text can be hidden using the CSS class usa-sr-only.
* Always a good idea to include a search box on the home page, especially if web site has a lot of pages
* Allow search bar to be wide as possible using a minimum of 27 characters wide.
* The magnifying glass is the best indicator icon for search.
* Be sure screen readers can read the search field.
* Include a search bar on the search results page.
* The search input field should be located on a home page.
* Don’t offer advanced search as default option.
* Be sure a label is present in the form field for screen reader users.

## Breadcrumbs

Breadcrumbs are links that appear in the top of the main content page to help visitors keep track of their location on the portal. CSS styles for the breadcrumb will be found in the tas-custom.css file.

|  |  |
| --- | --- |
|  | .va-nav-breadcrumbs--playbook a {  border-bottom: 2px solid #d6d7d9;  text-decoration: none ;  color: #0071bc;  }  .va-nav-breadcrumbs--playbook a:visited {  text-decoration: none ;  color: #0071bc;  }  .va-nav-breadcrumbs--playbook a:hover {  text-decoration: none;  background: rgba(0, 0, 0, 0.05);  border-bottom: 2px solid #fdb81e;  }  .va-nav-breadcrumbs--playbook li {  text-transform: none;  padding: .1em;  display: inline-block;  line-height: 1.15em;  font-size: .8em;  } |

## Modal Window/Alert messages

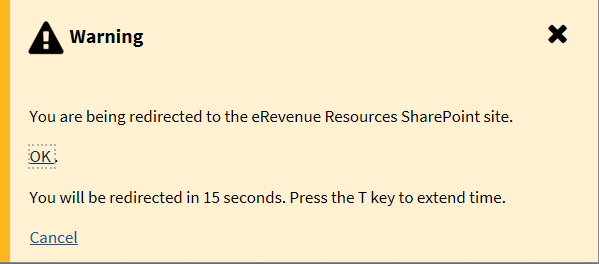
TAS has several types of modal windows and alert message

* External links.
* Modal windows.
* Error messages.

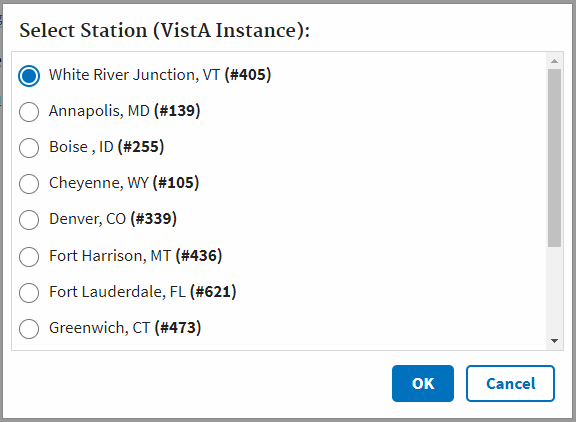
**External Links Alert Message**

The modal window is used for external links to let users know they will be navigating to an external site. The contents of the modal will contain a warning, display an icon and a dynamic counter. The counter counts down from 30 seconds to 0. When the counter reaches 0, the modal window will automatically close. If the user presses on the “T” in the keyboard, the timer will increase back to 20 seconds and start counting down again.

There is also an OK hyperlink – so if clicked on, the user navigates to an external site.

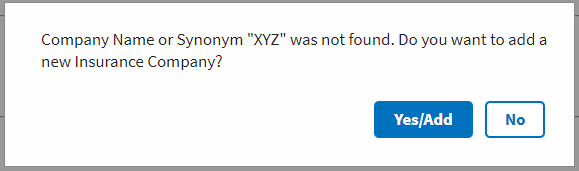


Modal window example with radio button list:



|  |  |  |
| --- | --- | --- |
| Title of Modal | Font family: Merriweather  Font size: 18pt  Font weight: bold | .modal\_header {  font-family: "Merriweather";  font-size: 18px;  font-weight: 700;  padding:8px;  } |
| Body content | Source Sans Pro, sans-serif  Font size: 1.7 rem  Font weight: normal  Line height: 1.5 rem | .modal\_body {  margin: 0% auto;  overflow-y: scroll;  padding-left: 10px;  height:300px;  border: 1px solid #d6d7d9;  } |
| Modal footer buttons | OK button always to the left of the Cancel button, right justified to the modal window | .modal\_footer {  text-align: right;  margin-top:5px;  width: 100% ;  } |

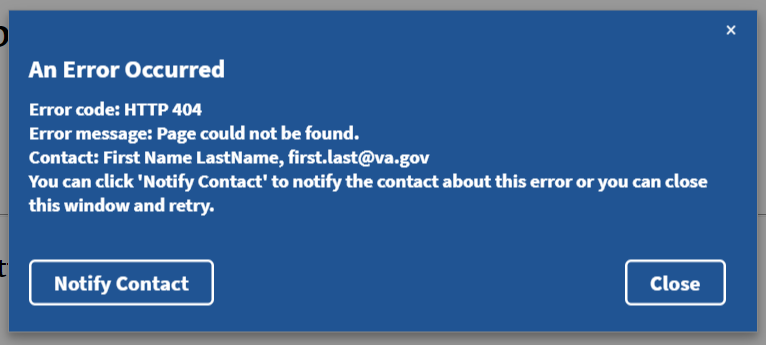
Modal window example with no title:



|  |  |  |
| --- | --- | --- |
| Body content | Font family: Source Sans Pro, sans-serif  Font-size: 1.7rem  Font-weight: normal | .modal\_content\_small {  height:auto;  overflow-y:auto;  padding-left:16px;  border: 0px solid #d6d7d9;  } |
| Modal footer buttons | Yes/Add button always to the left of the No or Cancel button, right justified to the modal window |  |

**Error Messages Modal Window**

If a user faces an error, a modal window will appear with error messages.



## Understanding Shared Components and Modules

A component can be used on different modules in the TAS application. Using Angular as the framework, shared modules in TAS organizes and streamlines code. Components share functionality and can be used on different web pages.

What is a shared module? Shared modules allow developers to use components in multiple modules, share instances of services and directives.

Step 1 Create a module.

Import {NgModule} from ‘@angular /core’;

Import {CommonModule} from ‘@angular/common’;

@NgModule ({

Imports: [ CommonModule ]

})

Export class DateModule {}

This module contains everything related to dates . A component called DatePickerComponent is included inside a module.

Import {NgModule} from ‘@angular /core’;

Import {CommonModule} from ‘@angular/common’;

@NgModule ({

Imports: [CommonModule ],

Declarations: [ DataPickerComponent]

})

Export class DateModule {}

To make the DatePickerComponent accessible for other modules, it must be added to the exports array:

Import {DatePickerComponent} from ‘./datepicker/component’;

Import {NgModule} from ‘@angular /core’;

Import {CommonModule} from ‘@angular/common’;

@NgModule ({

Imports: [ CommonModule],

Declarations : [ DataPickerComponent],

Exports: [DatePickerComponent]

})

Export class DateModule {}

Now the DatePickerComponent is available in other modules that import DateModule to have access to DatePickerComponent.

## Component File Structure

In the TAS file structure, many of the shared components, will be located inside src > app > core > mccf-common directory as shown below in the diagram.

|  |  |
| --- | --- |
|  | Each directory inside mccf-common directory is a shared component that consist of several file types:  HTML, CSS, TS and SPEC.TS  Services, directives and interfaces are also integrated with these components. |

## Tree View Component

TAS has incorporated the ngx-treeview component which can be found at the following link: <https://www.npmjs.com/package/ngx-treeview>

Features on Angular treeview

* Unlimited tree level
* State: Collapse, expand, disabled (indeterminate)
* Filtering
* Checkbox with tri-state

|  |  |
| --- | --- |
| Above is an example of an Angular treeview that displays 3 state checkboxes. | **EXAMPLE A HTML**  <tas-treeview [config]="config"  [items]="items"   (filterChange)="onFilterChange($event)"  (selectedChange)="values = $event">  </tas-treeview>  **TYPSCRIPT**  this.config = TreeviewConfig.create({  hasAllCheckBox: true,  hasFilter: true,  hasCollapseExpand: true,  hasHyperlink: false,  decoupleChildFromParent: false,  maxHeight: 1200  }) |

This component can display checkboxes in a treeview UI control that expands and collapses. Component has filtering capabilities, expand/collapse state, unlimited tree levels and optional expand/collapse all toggle represented by double arrows.

## 3 state checkboxes

|  |  |
| --- | --- |
| 3 state – example 1 | 3 state – example 2 |

There are 3 states to the checkbox: checked, unchecked and indeterminate. The last state is represented by a dash as shown in the screen grabs. In example 1, DIV4 checkbox is unchecked. VISN DEF is in the indeterminate state, so therefore its parent node CPAC North is also indeterminate. If DIV4 is checked as in example 2, then VISN DEF and CPAC North would be updated to checked. The parent node of the checkbox, VISN DEF will automatically update when it’s child node checkboxes are in both checked and unchecked state such as DIV3 and DIV4.

## EXPAND/COLLASPE TOGGLE

This toggle button will either expand all or collapse all nodes in one click.

|  |  |
| --- | --- |
| Collapse all state | Expand all stat |
|  |  |

## DEVELOPER NOTES:

To use the component, add the necessary HTML tags to the web page. Using TypeScript, the treeview has config parameters that are Boolean or string values that can modify the look of the treeview.

NOTE: Tree views are generated on a page load using recursive programming in Angular. The HTML markup will be different compared to the Example A HTML markup.

After page loads, recursive programming will change the HTML that will be composed of HTML tags including <DIVS>, <UL>, <LI> and <TAS-TREEVIEW-ITEM>.

This same component is also used on eInsurance web page as a side navigation UI control. The arrows will expand and collapse submenu items, however there are no checkboxes.

|  |  |
| --- | --- |
| **Angular HTML MARKUP** | **Sample Web Page MARKUP** |
| <tas-treeview  [config]="config"  [items]="items" (filterChange)="onFilterChange($event)" (selectedChange)="values = $event" >  </tas-treeview> | <ul class="tree" role="tree">  <tas-treeview-item>  <div class="treeview-item">  <li class="form-inline row-item" role="treeitem">  <i tabindex="0" class="fa fa-caret-right" id="caret\_CPAC North" aria-hidden="true"></i>  <div class="form-check">  <input tabindex="0" class="form-check-input p-default ng-untouched ng-pristine ng-valid" type="checkbox">  <label aria-selected="false" class="form-check-label" id="label\_CPAC North">CPAC North </label>  </div>  </li>  </div>  </tas-treeview-item>  <tas-treeview-item>  …  </tas-treeview-item>  <ul> |

|  |  |
| --- | --- |
|  | **HTML**  <div id="sidenav">  <tas-treeview [config]="config" [items]="items" >  </tas-treeview>  </div>  **TYPESCRIPT**  this.config = TreeviewConfig.create({  hasAllCheckBox: false,  hasFilter: false,  hasCollapseExpand: true,  maxHeight: 1200  })  } |

|  |
| --- |
| <div class="treeview-container" style="max-height: 1200px;">  <tas-treeview-item>  <div class="treeview-item">  <li class="form-inline row-item" role="treeitem">  <i aria-expanded="true" aria-hidden="true" class="fa fa-caret-down" tabindex="0" id="caret\_Reports">  </i>  <div class="form-check">  <label aria-selected="false" class="form-check-label" id="label\_Reports">Reports  </label>  </div>  </li>  <div>    <tas-treeview-item>  <div class="treeview-item">  <li class="form-inline row-item" role="treeitem">  <i aria-expanded="true" aria-hidden="true" class="fa fa-caret-down" tabindex="0" id="caret\_Claim Results and Status">  </i>  <div class="form-check">  <label aria-selected="false" class="form-check-label" id="label\_Claim Results and Status">  Claim Results and Status  </label>  </div>  </li>  <div>  <tas-treeview-item>  <div class="treeview-item">  <li class="form-inline row-item" role="treeitem">  <div class="form-check">  <a role="presentation" href="/epharmacy/sidenav/common/help"> Totals By Date </a>  </div>  </li>  <div>  </div>  </div>  </tas-treeview-item>  <tas-treeview-item>  <div class="treeview-item">  <li class="form-inline row-item" role="treeitem">  <div class="form-check">  <a role="presentation" href="/epharmacy/sidenav/einsurance/home"> Recent Transaction </a  </div>  </li>  <div>  </div>  </div>  </tas-treeview-item>  </div>  </div>  </tas-treeview-item>    <tas-treeview-item>  <div class="treeview-item">  <li class="form-inline row-item" role="treeitem">  <i aria-expanded="true" aria-hidden="true" class="fa fa-caret-down" tabindex="0" id="caret\_Other Reports">  </i>  <div class="form-check">  <label aria-selected="false" class="form-check-label" id="label\_Other Reports">  Other Reports  </label>  </div>  </li>  <div>    <tas-treeview-item>  <div class="treeview-item">  <li class="form-inline row-item" role="treeitem">  <div class="form-check">  <a role="presentation" href="/epharmacy/sidenav/epharmacy/home"> Payer Sheet Detail Report </a>  </div>  </li>  <div>  </div>  </div>  </tas-treeview-item>  <tas-treeview-item>  <div class="treeview-item">  <li class="form-inline row-item" role="treeitem">  <div class="form-check">  <a role="presentation" href="/epharmacy/sidenav/non-mccf/home"> ECME Setup = Pharmacies Report </a>  </div>  </li>  <div>  </div>  </div>  </tas-treeview-item>  </div>  </div>  </tas-treeview-item>  <tas-treeview-item>  <div class="treeview-item">  <li class="form-inline row-item" role="treeitem">  <i aria-expanded="true" aria-hidden="true" class="fa fa-caret-down" tabindex="0" id="caret\_COB Reports">  </i>  <div class="form-check">  <label aria-selected="false" class="form-check-label" id="label\_COB Reports">  COB Reports  </label>  </div>  </li>  <div>  <tas-treeview-item>  <div class="treeview-item">  <li class="form-inline row-item" role="treeitem">  <div class="form-check">  <a role="presentation" href="/epharmacy/sidenav/cobreporta"> COB Report A </a>  </div>  </li>  <div>  </div>  </div>  </tas-treeview-item>  <tas-treeview-item>  <div class="treeview-item">  <li class="form-inline row-item" role="treeitem">  <div class="form-check">  <a role="presentation" href="/epharmacy/sidenav/cobreportb"> COB Report B </a>  </div>  </li>  <div>  </div>  </div>  </tas-treeview-item>  </div>  </div>  </tas-treeview-item>  </div>  </div>  </tas-treeview-item>  </div> |

## Dynamic Select box

Dropdown menu are also shared components. Using configService in TypeScript, JSON data can be used to populate the options and values of the select box.

|  |  |
| --- | --- |
|  | **HTML**  <tas-dropdownmenu id="ddm1"></tas-dropdownmenu>  **TYPESCRIPT**  this.configService.load(["menuoptions", "selectBoxOptions"])  .subscribe(config => {  this.myoptions = config.selectBoxOptions  let selectshape = (document.getElementById("shapes")) as HTMLSelectElement;  for (let j = 0; j < this.myoptions.length; j++) {  let opt = document.createElement("option");  opt.value = j.toString();  opt.text = this.myoptions[j];  selectshape.appendChild( opt )  }  }) |

## Date Picker

Date picker is another Angular shared component that can easily be configured to display past and future dates. Multiple date pickers can be displayed on a page.

|  |  |
| --- | --- |
|  | **HTML**  <tas-date-picker id="begin"></tas-date-picker>  **TYPESCRIPT**  @Component({  selector: "tas-date-picker",  exportAs: "datepicker",  styles: [dpStyles],  template: dpTpl,  providers: [UtilService, DP\_VALUE\_ACCESSOR],  encapsulation: ViewEncapsulation.None  }) |
|  | Date picker Alert Box  If a user clicks the Open Report button without selecting a dates from either date picker control, then an Error alert box will appear on the page with a message saying both date fields must have values. |

## Quick Links

Quick Links are hyperlinks that point to external systems outside of the TAS application.

|  |  |
| --- | --- |
|  | **HTML**  <tas-quick-links [extLinks]="extLinks"></tas-quick-links>  **TYPESCRIPT**  import { Component, Input } from '@angular/core';  import { ExtLink } from '../../core/ext-link/ext-link'  @Component({  selector: 'tas-quick-links',  templateUrl: './quick-links.component.html',  styleUrls: ['./quick-links.component.css']  })  export class QuickLinksComponent {  @Input() extLinks: Array<ExtLink>  constructor() { }  } |

## Tooltips

Tooltips is a pure CSS library where a bubble/hint appears on a mouse hover over web page elements such as links and graphics. All the CSS classes are located in tas-custom.css file called.hint Bubbles can appear in any desired direction such as top, bottom, left, right. This feature uses no JavaScript, it supports accessibility and works in all the major browsers.

|  |  |
| --- | --- |
|  | **HTML**  <span class="hint--top hint--rounded" aria-label="Contact Us">Contact Us</span>  **CSS**  [class\*="hint--"]:after {  text-shadow: 0 -1px 0px black;  box-shadow: 4px 8px rgba(0, 0, 0, 0.3);  } |

## Pagination

Pagination works by presenting a set number of rows in a view, with the ability to navigate to another set. Break a complete dataset into smaller sequential parts and provide separate links to each. Provide pagination control to browse from page to page. Let the user browse to the previous and next pages by providing links to such actions. Also, provide links to the absolute start and end of the dataset (first and last).

Pagination provides the user with a natural break from reading or scanning the contents of the dataset and allows them to re-evaluate whether they wish to continue looking through more data, or navigate away from the page. This is also why pagination controls are most often placed below the list: to provide the user with an option to continue reading through the larger dataset.

## Usability

* Use when it is unsuitable to display all the data on a single page/screen.
* Use when the dataset is in some way ordered.
* Do not use when you don’t want the user to pause for navigating to the next page.

## Accessibility

* User should be able to navigate by keyboard.
* User should be able to know that there is a pagination navigation when scanning the page with a screen reader.
* Each navigation item should be read as "Goto Page 1".
* Navigation should have Next and Previous links.

|  |
| --- |
| <div class="paginator">  <taspaginationtemplate>  <div class="ngx-pagination" role="navigation" aria-label="Pagination">  <button class="pagination-previous disabled" aria-label="Previous page">  <span>Previous <span class="show-for-sr">page</span>  </span>  </button>  <button class="small-screen">  1 / 2  </button>  <button id="pageIndexBtn1" class="current">  <span>  <span class="show-for-sr">You're on page 1 </span>  <span>1</span>  </span>  </button>  <button id="pageIndexBtn2">  <span>  <span class="show-for-sr">page </span>  <span>2</span>  </span>  </button>  <button class="pagination-next" aria-label="Next page">  <span> Next <span class="show-for-sr">page  </span>  </span>  </button>  </div>  </taspaginationtemplate>  </div> |
|  |

## Tableau Reporting

Tableau is a data visualization software that is used for data science and business intelligence. Tableau can create a wide range of different visualizations to interactively present the data and showcase insights. As an analytics platform, Tableau authors can design charts or reports to analyze data to provide better insight on the business.

## Usability

Tableau is a flexible and creative platform. Authors play a key role in designing charts that meet accessibility guidelines. Color choices, input choices, positioning of content, and textual descriptions all need consideration when building an accessible chart. To meet accessibility standards, charts should be designed to include views that are accessible to users who use screen readers and key-board navigation. Authors can publish these views that are embedded in a web page that conforms to the Web Content Accessibility guidelines. The chart should be clean, uncluttered and easy to navigate.

Tableau platform resolved many of the underlying technical challenges for accessibility compliance such as keyboard support, ARIA roles, and focus indicators. However, authors still need to compose a chart that comply to the Web Content Accessibility guidelines (WCAG 2.0 AA). When assigning colors to different dimension values, make sure that they provide enough contrast and aren't too close to each other on the light-dark spectrum.

Tableau also supports the following features to help create accessible views:

* Keyboard navigation
* Programmatic context for assistive technologies (using ARIA roles)
* A text equivalent for charts and visualizations
* Ability to conform with contrast standards
* Authentication when signing in to Tableau Server for embedded views

Support for accessibility standards helps enhance the mission by making content produced with Tableau software accessible to those living with various impairments. Below is a good example of a chart that would pass 508 compliance. Each section is described in greater detail in this guide. When building views, keep these 4 items in mind:

1. Keep it simple
2. Show more text and make it helpful
3. Use of color-blind palette and provide contrast
4. Provide other visual cues such position, size and shape

|  |  |
| --- | --- |
|  | A) Titles and Captions  B) View Data Page  C) Single/multi value filters  D) Color-blind palette |
|  | **View Data Page**  Page containing all the data for the rows and columns of the chart. |

## Accessibility

Tableau authors can create WCAG conformant view following these general steps:

STEP 1: Author views in Tableau Desktop and follow best practices

STEP 2: Publish view to Tableau Server or Tableau Online

STEP 3: Test the views using keyboard navigation and a screen reader.

Most views that are created following the best practice guidelines in Tableau Desktop will be WCAG conformant. They are treated by screen readers as images with a title and description. Users will be able to navigate using the keyboard to open the View Data window and access underlying data for the view. It is important to use text more liberally in the view to provide context for the view and its marks.

**View controls that support accessibility**

Authors will need to use the Tableau Desktop to create a view that include WCAG elements and then publish and embed that view to a web page.

**Best Practices for Designing Accessible Views**

**Perceivable –** chart information and component must be presented to users in a way that they can perceive. This means using text alternatives, and alternate ways to present content.

**Operable –** components should have easy navigation to users on different devices or methods that use to interact with the view.

**Understandable** **–** information in the view should be understandable. This means using distinguishable names and labels for different elements in the views.

* **KEEP IT SIMPLE** **–** if you have a lot of information that is present in the view, too much information will be difficult for users to understand or navigate. Use the following guidelines to keep views easy to understand without too much clutter of information.
  1. **Aggregate data whenever possible to help reduce the number of marks.** If there are more than 1000 marks in a view, the server renders the view instead of the browser and server rendered views are not WCAG compliant.
  2. Users have access the **View Data Page** (enabled by default) to review underlying data for the marks or they can download the data from the page to an accessible application to view it that way.

This is an example of two views to illustrate difference between detailed and aggregate views.

|  |  |
| --- | --- |
| More accessible - Aggregated view | More accessible - Aggregated view |
| https://onlinehelp.tableau.com/current/pro/desktop/en-us/Img/notaccessible_5000marks.png | https://onlinehelp.tableau.com/current/pro/desktop/en-us/Img/accessibilitycompliant_aggregated.png |
| Too much detail in this view – makes it harder to understand.  View contains over 5000 marks and not enough text to explains what the marks represent.  View is too difficult for users who use screen readers. | This view has data aggregated at a higher level.  Key data points are included and easiers for users to read and understand.  Marks reduced from 5000 to 20.  Users can read underlying details for the marks by placing focus on the any of the bar columns, press enter to open **View Data** page. |

* **USE SIMPLE GRAPHIC ELEMENTS** such as bar/line charts where text and color can be used.
* **LIMIT THE NUMBER OF MARKS to only emphasize the more important data points.**

Titles and captions for each view should provide descriptive text to make it more perceivable and understandable. This allows screen readers to read content in descriptive detail. Use text in titles and captions to describe data visualizations and avoid using acronyms or abbreviations. Don’t include works like “image of” or “picture of” in text descriptions because screen readers sometimes include this information. Avoid using all capital letters in heading – they can be difficult to read.

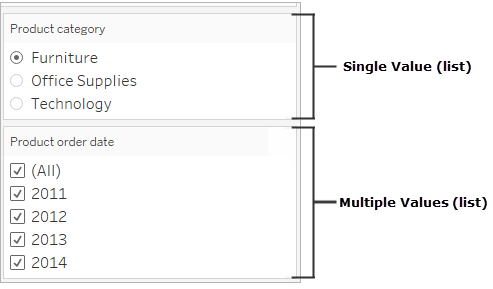
|  |  |
| --- | --- |
| Not easily accessible | More accessible – adding descriptive text to provide context |
| https://onlinehelp.tableau.com/current/pro/desktop/en-us/Img/accessibility_NoTitlesandCaptions.png | https://onlinehelp.tableau.com/current/pro/desktop/en-us/Img/accessibility_MoreTitlesandCaptions.png |
| A one-word title is not enough.  There is no caption or explanatory text to help explain this view. | This view has more text in the title and caption.  Each bar has a label on the top which provides more context. |

* **ADDITIONAL TEXT** 
  + Use text in **headings** on **legends** or filters to describe controls and what it does.
  + Show titles for the worksheet, filters and legends.
  + Refer to controls by labels whenever possible.
  + Avoid words like “Click here” or “More” or “More information”. These examples are too generic and can be confusing to users.
  + Use text in **captions** to provide a summary of what is shown in the view.
  + Use **labels** – by default these are not shown in a view so it is important to include them.
  + Be sure to add mark labels to display numbers so users do not have to rely just on colors alone to understand the data.

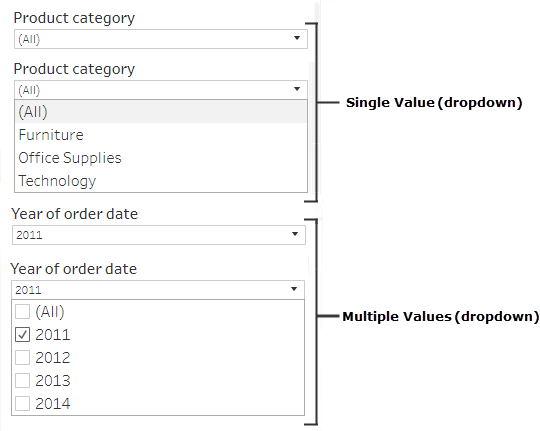
Below are screen shots from Tableau on how to make the charts more accessible:

|  |  |  |
| --- | --- | --- |
| https://onlinehelp.tableau.com/current/pro/desktop/en-us/Img/access_showtitlescaptions.png | https://onlinehelp.tableau.com/current/pro/desktop/en-us/Img/access_showmarklabels.png | https://onlinehelp.tableau.com/current/pro/desktop/en-us/Img/access_editcaption.png |
| Adding title and caption to worksheet | How to display labels. | Adding text to captions. |

* **USE FILTERS TO REDUCE NUMBER OF MARKS**
  + Using **filters** helps to focus the number of marks in a view to only what users need to see.
  + The filter modes that currently support WCAG conformance in Tableau are:
    - Single value list: use of radio buttons
    - Multiple value list: list of items with checkboxes



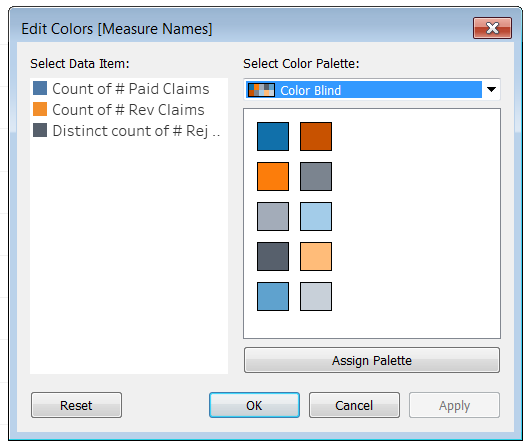
Drop down menu for filters



* **COLOR AND CONTRAST**
  + Color can help users distinguish marks in a view, but it is advisable to use a palette that has enough contrast to assist those with color-blindness.
  + Tableau has a built-in colorblind-friendly palette that should be used when creating any type of data visualization. This palette works very well for people with color-blindness. Below is the default Tableau colorblind-friendly palette.
  + For line marks, use additional options like shapes, size and labels.
  + Make sure color contrast for text is strong enough to meet contrast ratio standards of 4.5:1.

|  |  |
| --- | --- |
| Not easily accessible- using only color to for each line graph. | More accessible since it uses color and shape to differentiate marks. |
| https://onlinehelp.tableau.com/current/pro/desktop/en-us/Img/accessibility_badcolor.png | https://onlinehelp.tableau.com/current/pro/desktop/en-us/Img/accessibility_colorshape.png |
| View uses color for each line graph.  Marks in this view are not using a color palette that is fully accessible to users with visual impairments. | Uses both color and shape to identify marks.  Color and shape legend with titles identify color and shapes represented in the view. |

It is good practice to use Tableau’s built-in **color-blind palette**.



**Important**: When you use color in the view, you must also provide other types of encoding for marks, such as labels, size, shape, and position. Don't rely on color alone to communicate differences in marks.

**Publishing views**

To make views available to users, publish view to Tableau Server or Tableau Online. In the toolbar menu, click **Share** to push out changes to the server. According the TAS business requirements, all data views will be viewed from a new IE tab in the browser outside of the TAS application.

**Testing published views**

* When creating a view, test browser zoom to 200% to make sure all the components in view are displayed as intended
* Avoid positioning filters horizontally in a worksheet. There is chance that components such as the filters can overlap with visualization.
* Don’t reposition filters after you add them to view. Repositioning filters or change in the order may cause the tab order to change.
* Test the view using the **keyboard shortcuts**. See below for more details.

**KEYBOARD ACCESS FOR VIEWS**

|  |  |
| --- | --- |
| All | * **Tab**: Move focus to next component * **Shift + Tab** Move focus to previous component |
| Sheet Tabs | * **Left/Up**: Move focus to previous * **Right/Down**: Move focus to next * **Enter**: Navigate to the focused tab |
| Visualization (includes the view area, filters, and legends) | Legends   * **Up**: Move focus to previous * **Down**: Move focus to next * **Left**: (Single column) move focus to previous, (Multi-column) move focus to left * **Right**: (Single column) move focus to next, (Multi-column) move focus to right * **Home**: Move focus to first item * **End**: Move focus to last item * **Space**: Toggle item selection * **Enter**: Toggle highlighting on and off * **Esc**: Clear all item selections   Filters   * **Left/Up**: Move focus to previous item * **Right/Down**: Move focus to next item * **Home**: Move focus to first item * **End**: Move focus to last item * **Space**: Toggle selection (check/clear checkboxes, select/ clear radio buttons) * **Enter**: Apply changes, if **Cancel** or **Apply** buttons available * **Esc**: Revert changes to default settings.   Filter controls and search   * **Shift +Tab**: Move focus from filter item to filter control buttons (search, Single Value (List)/Multiple Values (List), Single Value (Dropdown)/Multiple Values (Dropdown), Include/Exclude Values). Press **Space** to click a button. * For the drop-down menu, use **arrow keys** to move focus. Press **Enter** to choose a menu item. * For search, press **Space** on the Search button, and then enter search terms. Press **Esc** once to clear input. Press **Esc** again to dismiss Search. |
| View Data window | * **Esc** Clear mark selections. * When the view has the focus, press **Enter**, or press **Ctrl**+ **Shift**+ **Enter** to show the View Data window. The screen reader you are using might require some combination of modifier keys (**Shift**, **Ctrl**, **Alt**, **Cmd**) in conjunction with **Enter**. See the documentation for your screen reader.   In the View Data window (when screen reader is on):   * **Arrow keys**: Move within table. * **Tab**: Move focus to download data link in the window. Press **Enter** to navigate to link. |

# Appendix A – Wireframes

Figure 1: TAS Wireframe – eBilling

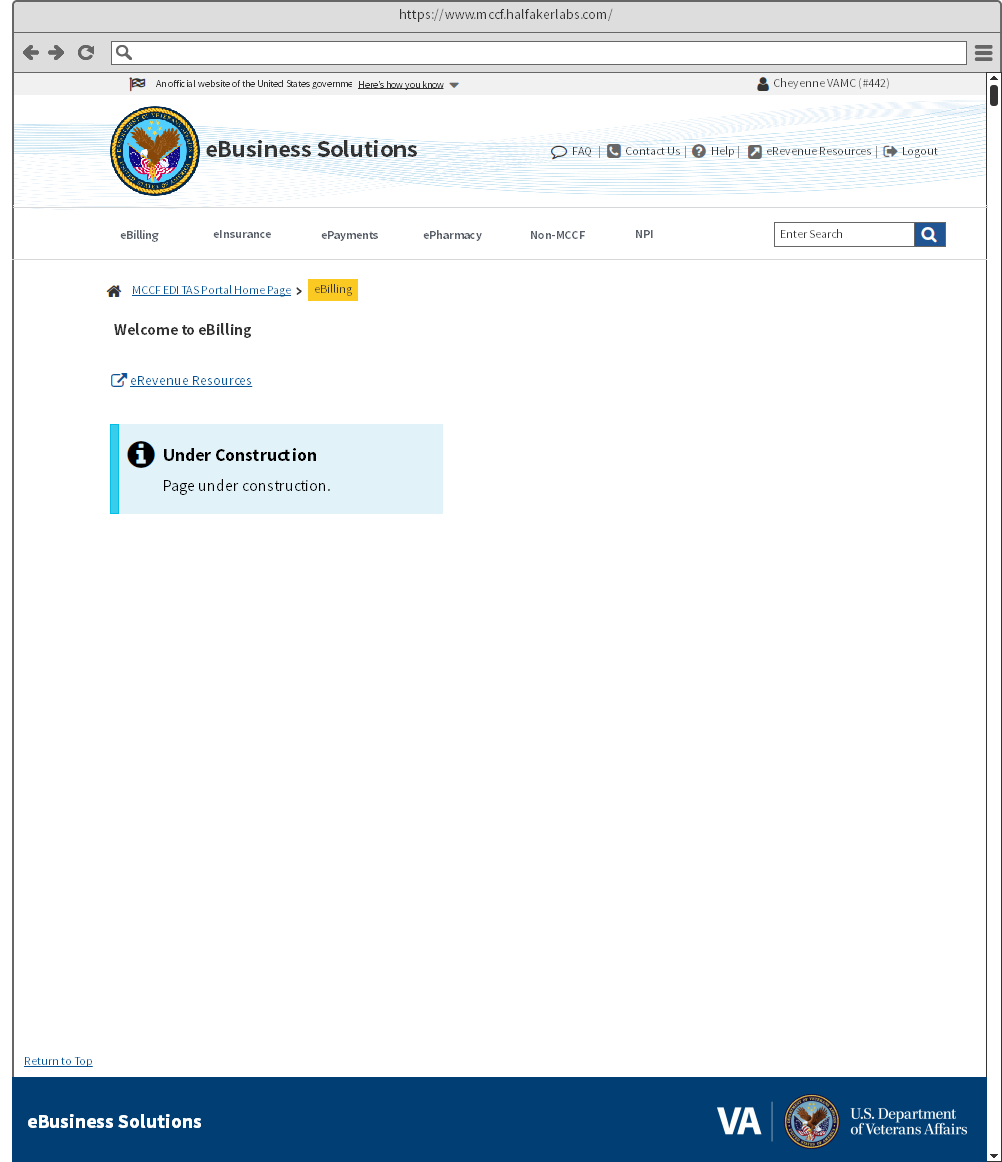


Figure 2: TAS Wireframe eInsurance

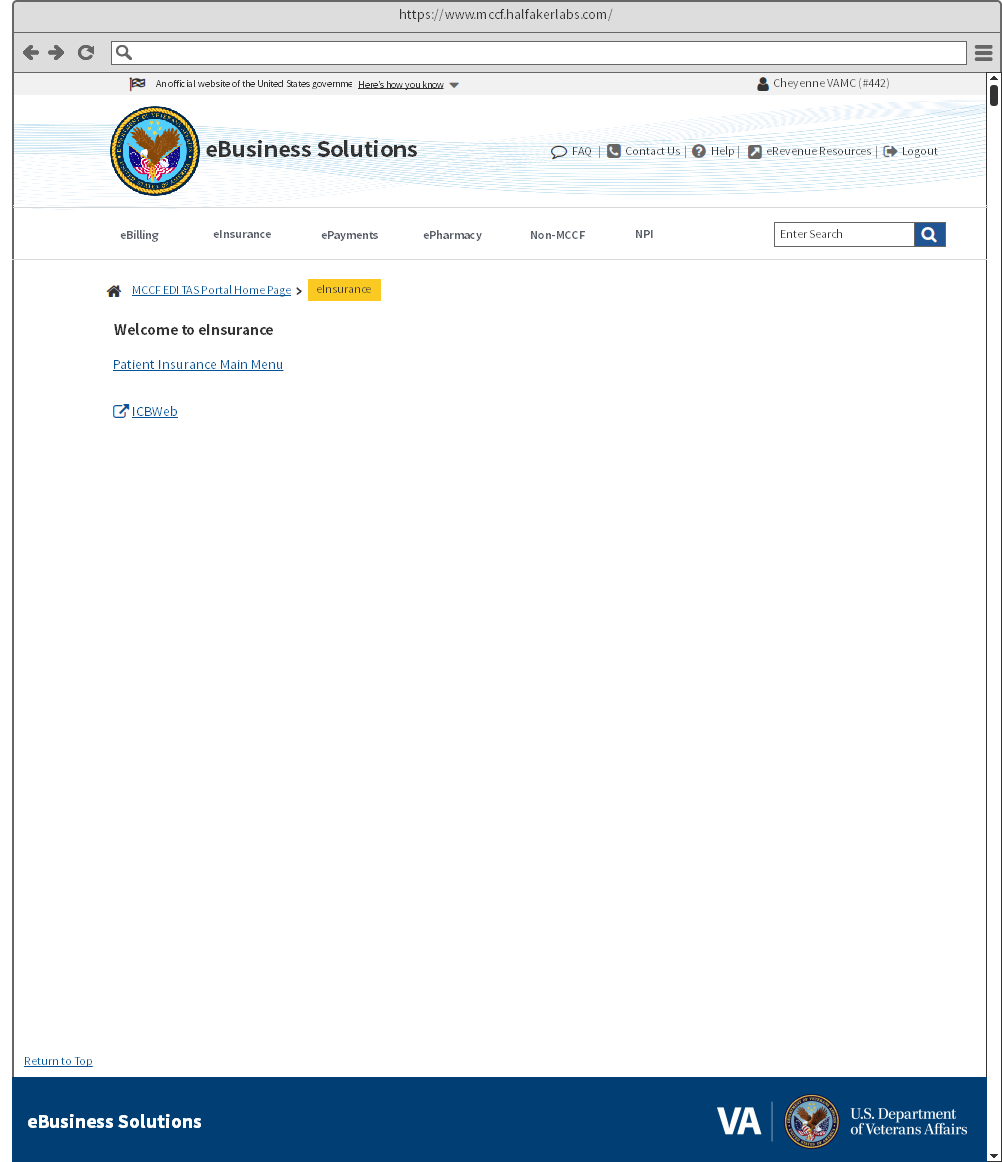


Figure 3: TAS Wireframe eInsurance - Patient Insurance Main Menu

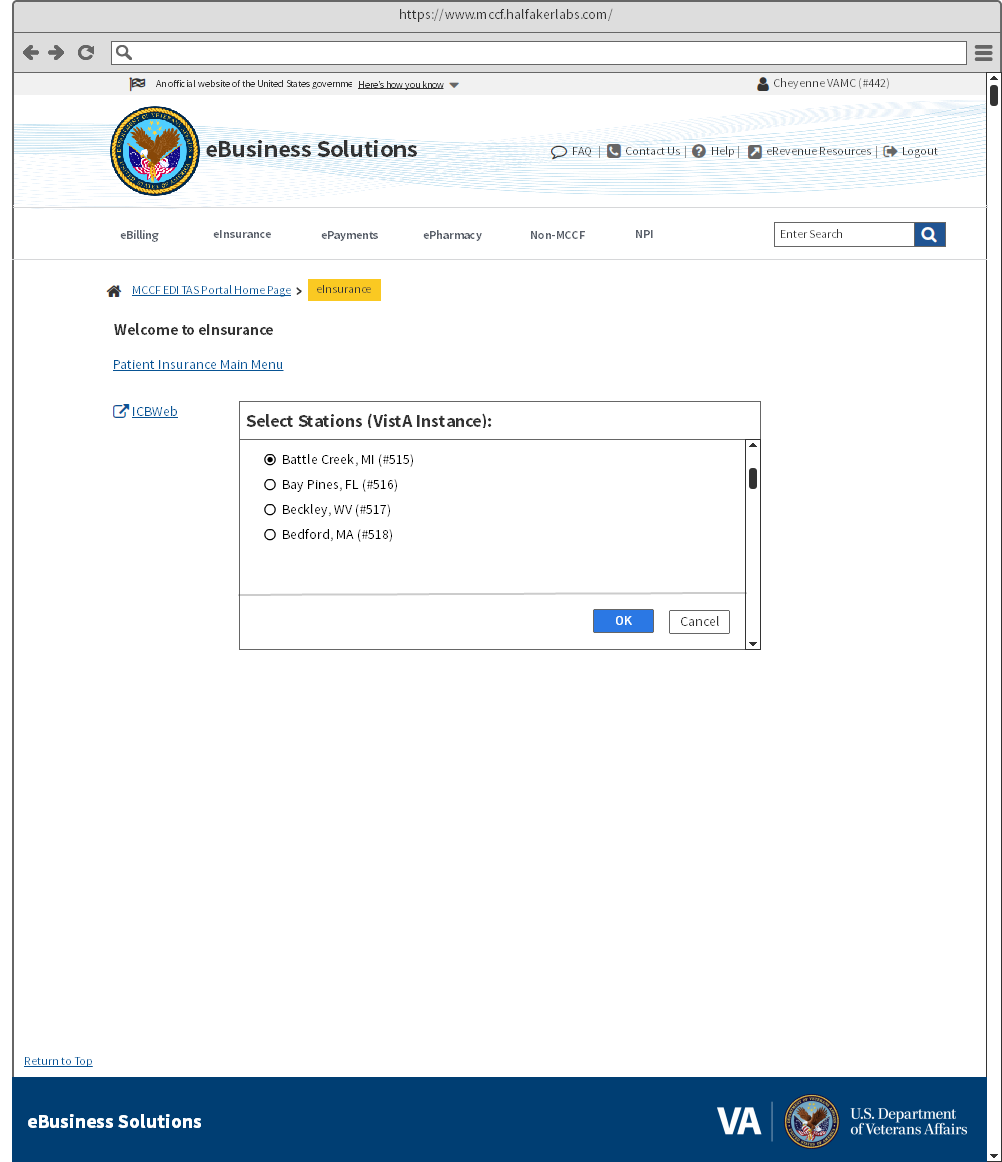


Figure 4: TAS Wireframe Payments

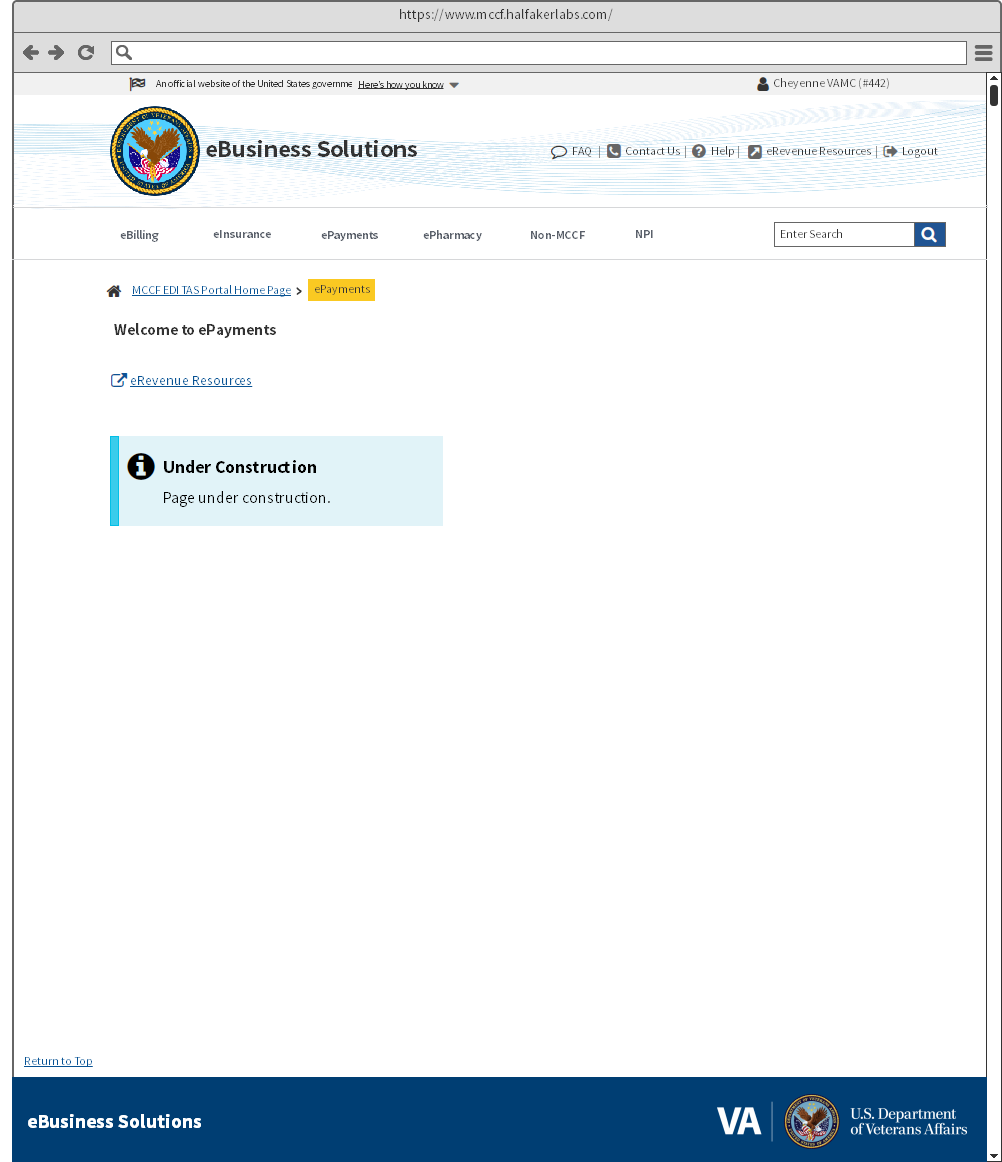


Figure 5: TAS Wireframe ePharmacy

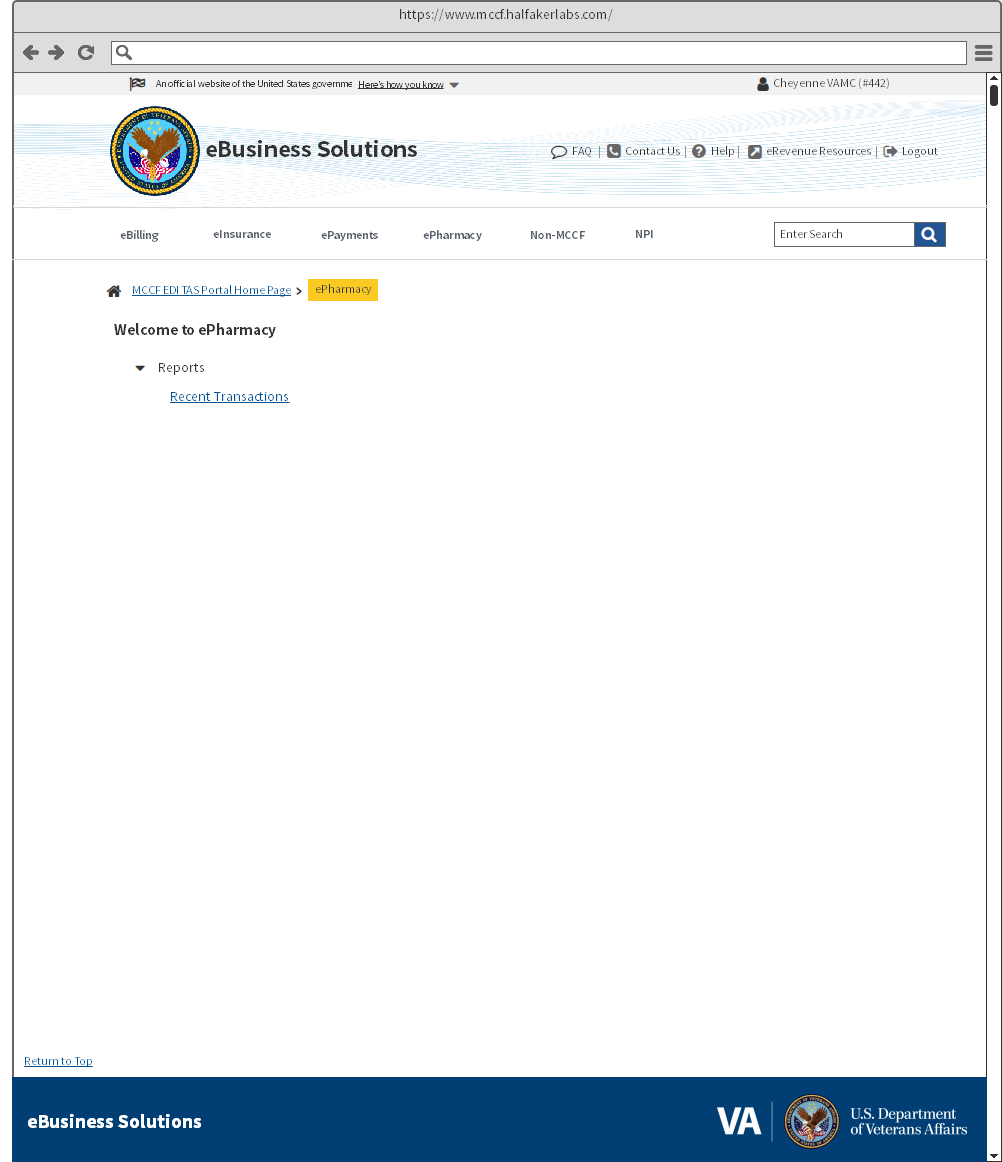


Figure 6: TAS Wireframe ePharmacy Recent Transactions

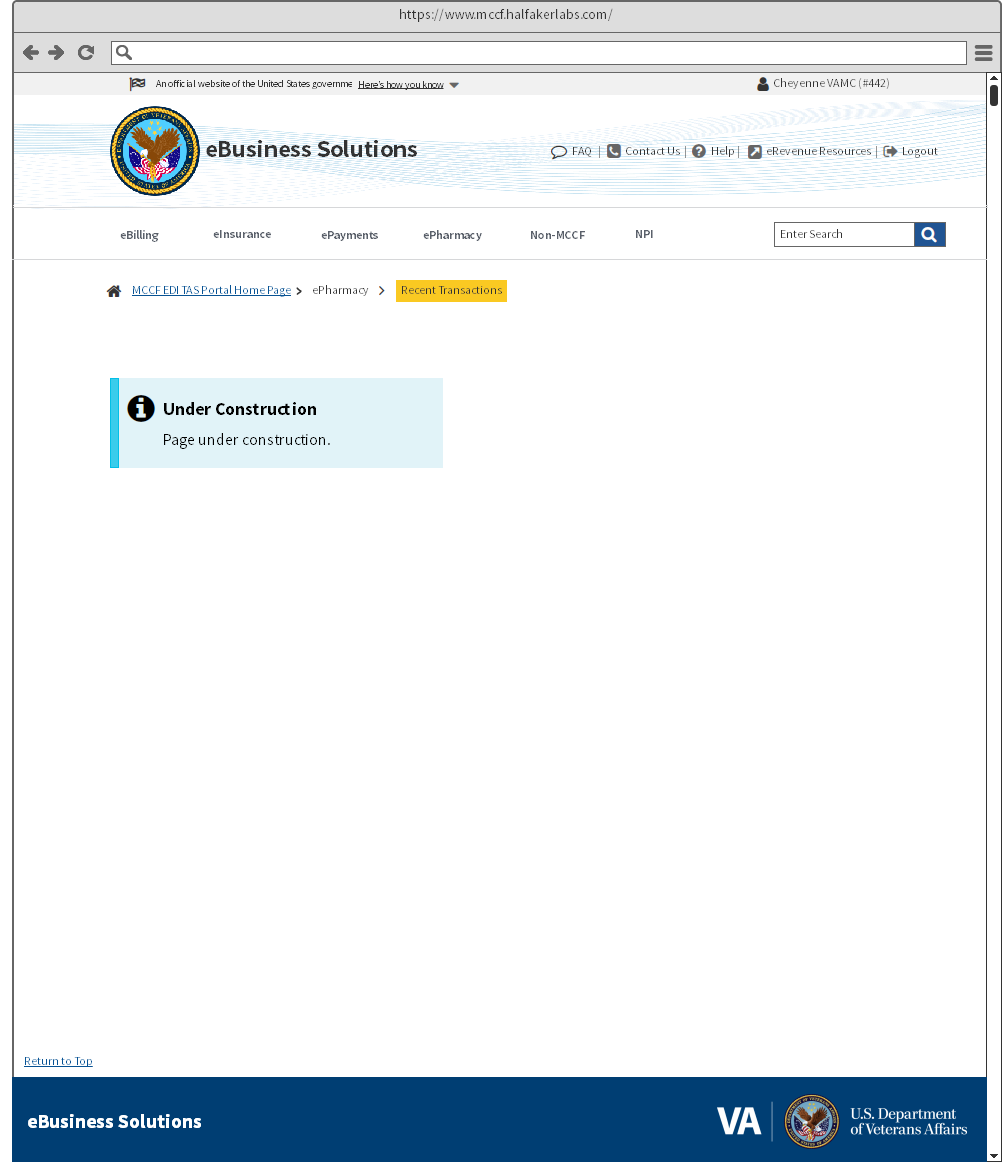


Figure 7: TAS Wireframe Non-MCCF

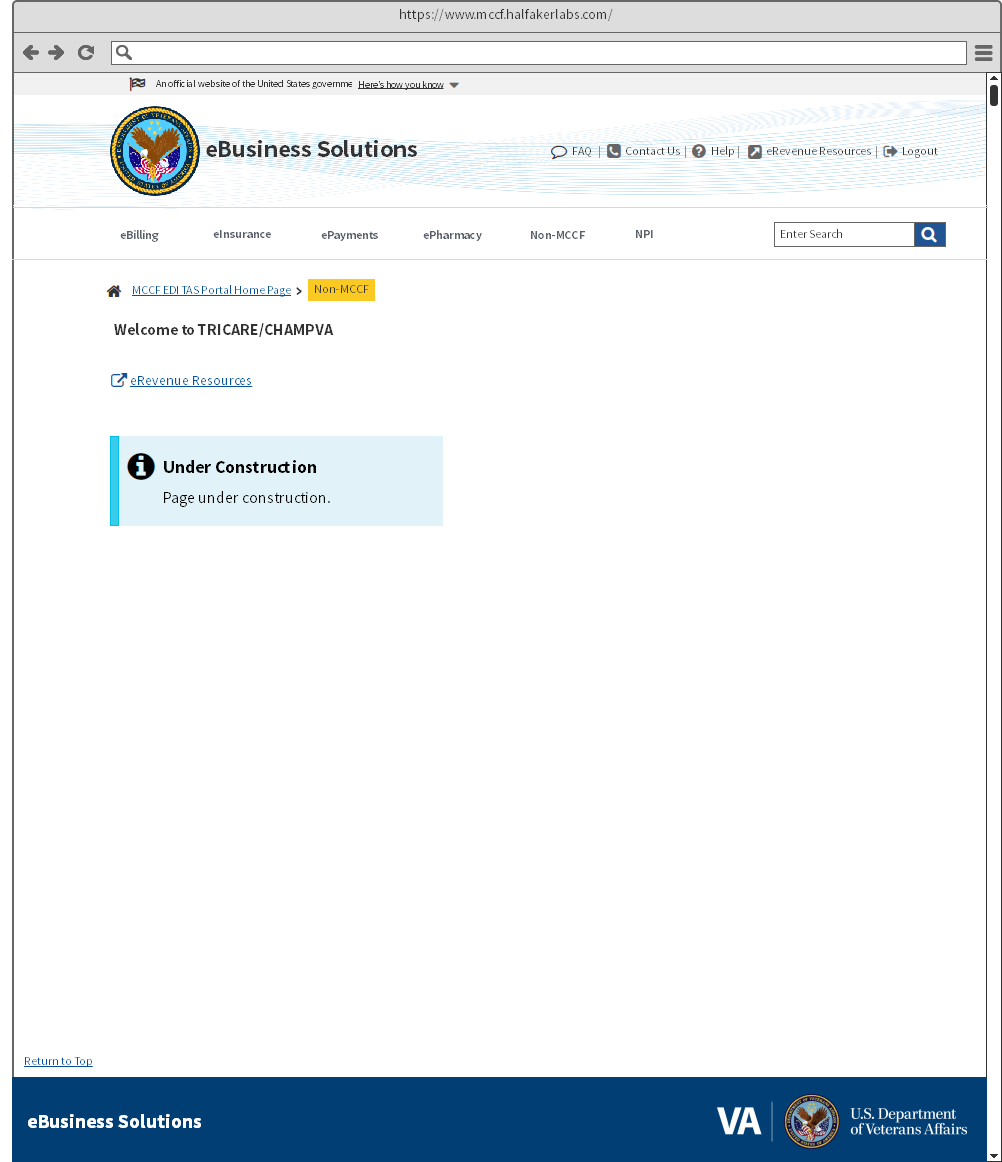


Figure 8: TAS Wireframe NPI

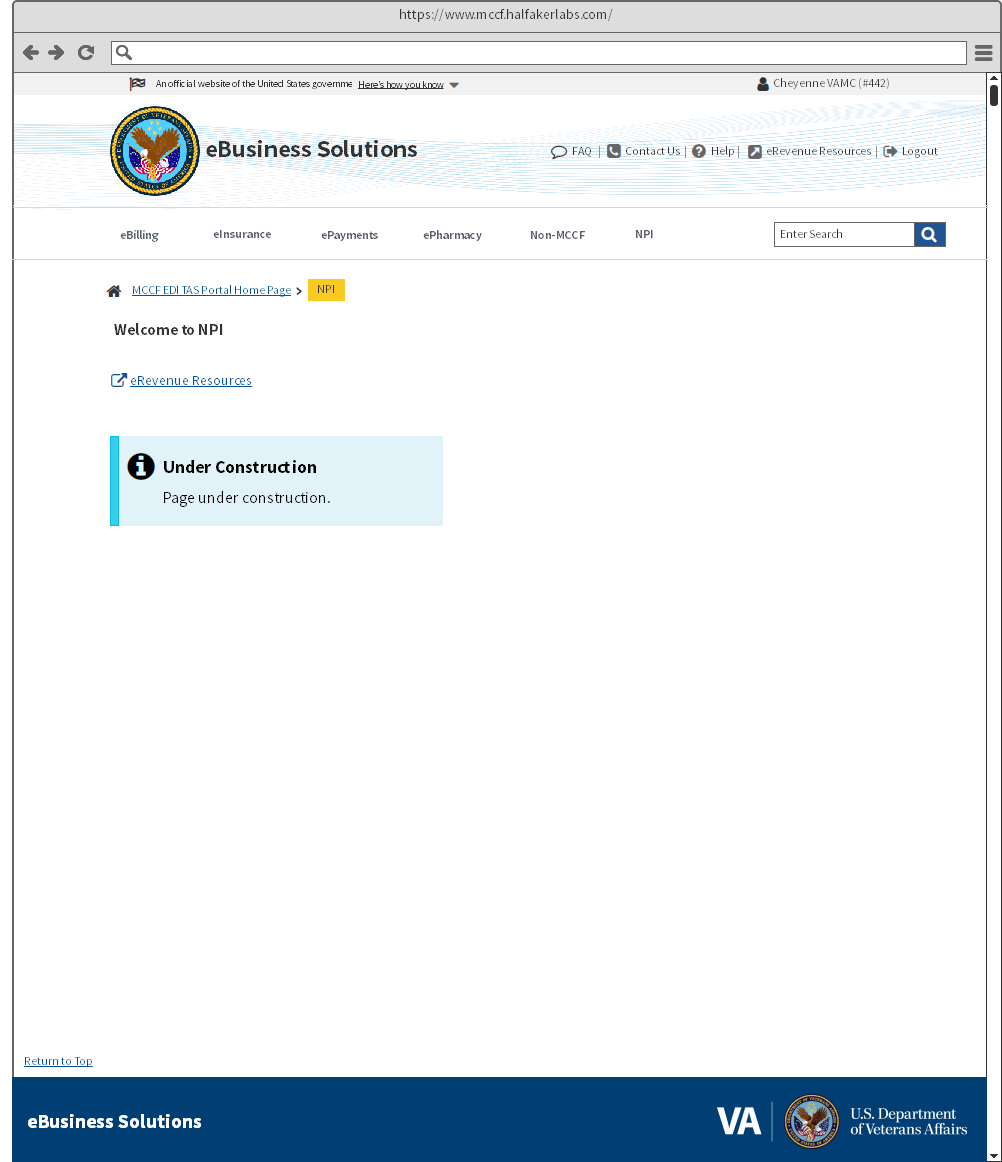


Figure 9: TAS Wireframe FAQ

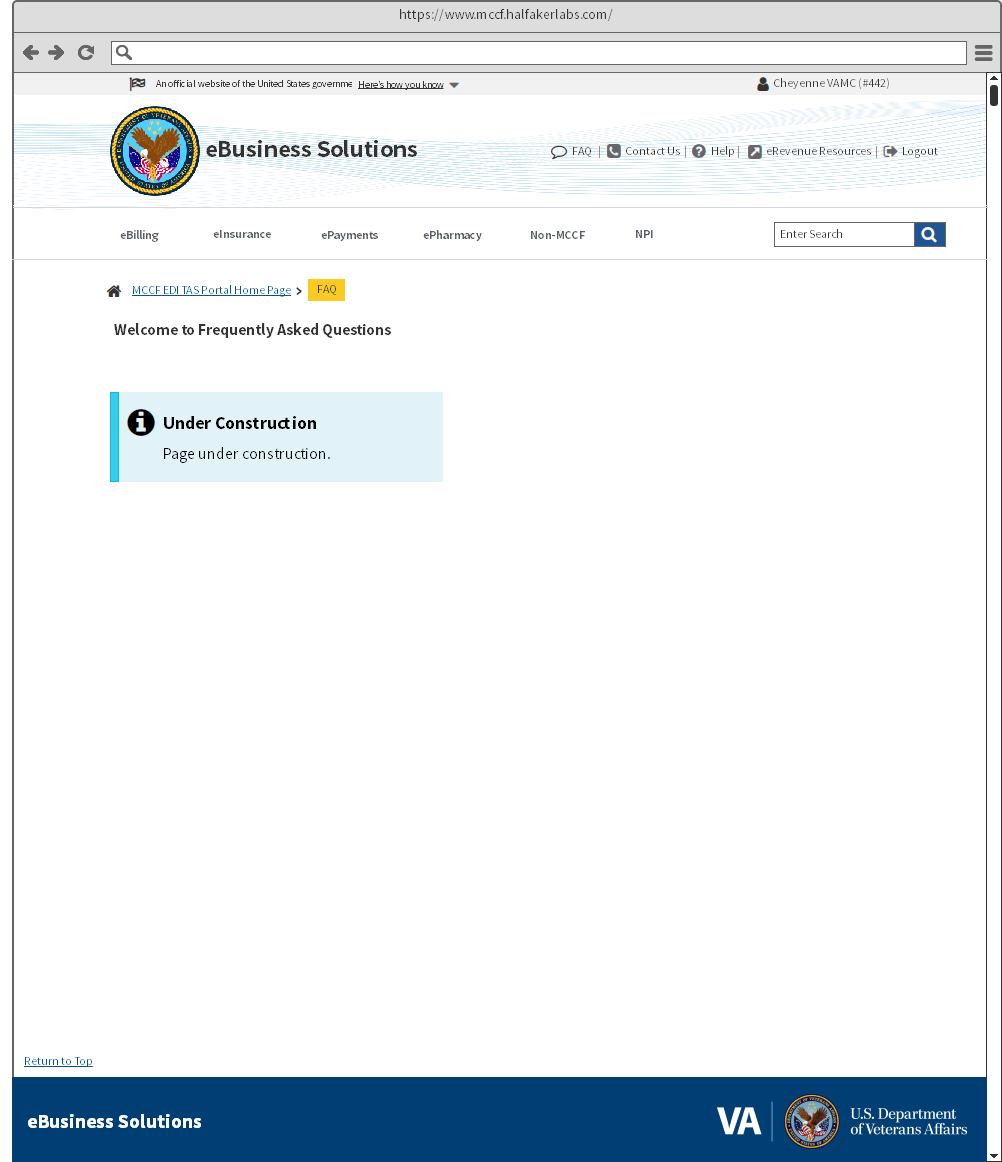


Figure 10: TAS Wireframe Contact Us

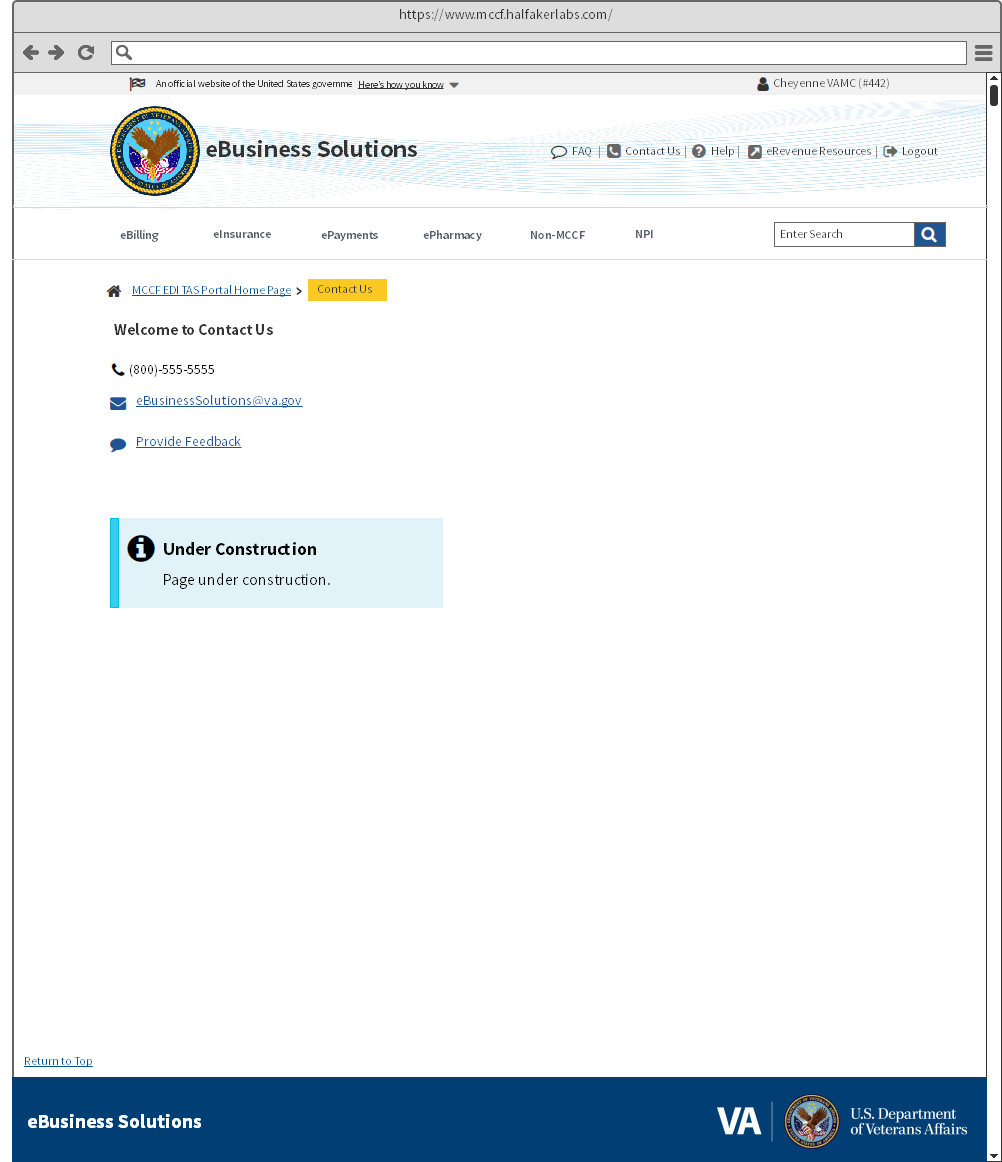
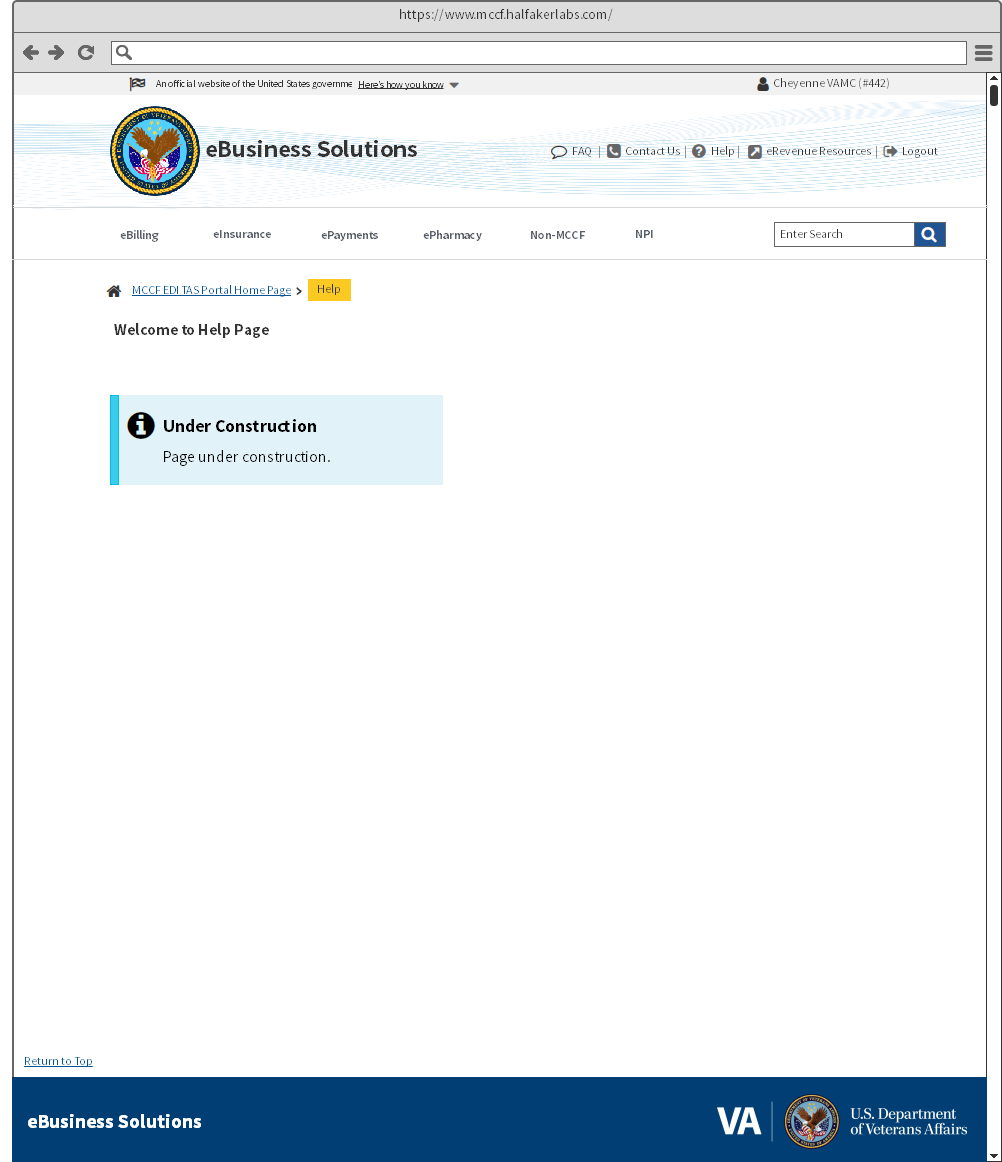


Figure 11: TAS Wireframe Help



* 1. Template Revision History

| **Date** | **Version** | **Description** | **Author** |
| --- | --- | --- | --- |
| January 2018 | 1.0 | Initial Draft | TASCore Design Team |
| July 2018 | 1.1 | Shared Components | TASCore Design Team |
| September 2018 | 1.2 | Added Wireframe Appendix | TASCore Design Team |
| November | 1.3 | Update data table | TASCore Design Team |