**MCCF EDI TAS US1406 - US1409**

System Design Document

TASCore Build #TBD



Department of Veterans Affairs

**June 2017**

Version 1.0

**User Story Number:** US1406/US1409

**User Story Name:** Define Layout of the MCCF EDI Home Page/Product Landing Page

To satisfy the enhancement request described by the aforementioned User Story, the following need to be completed:

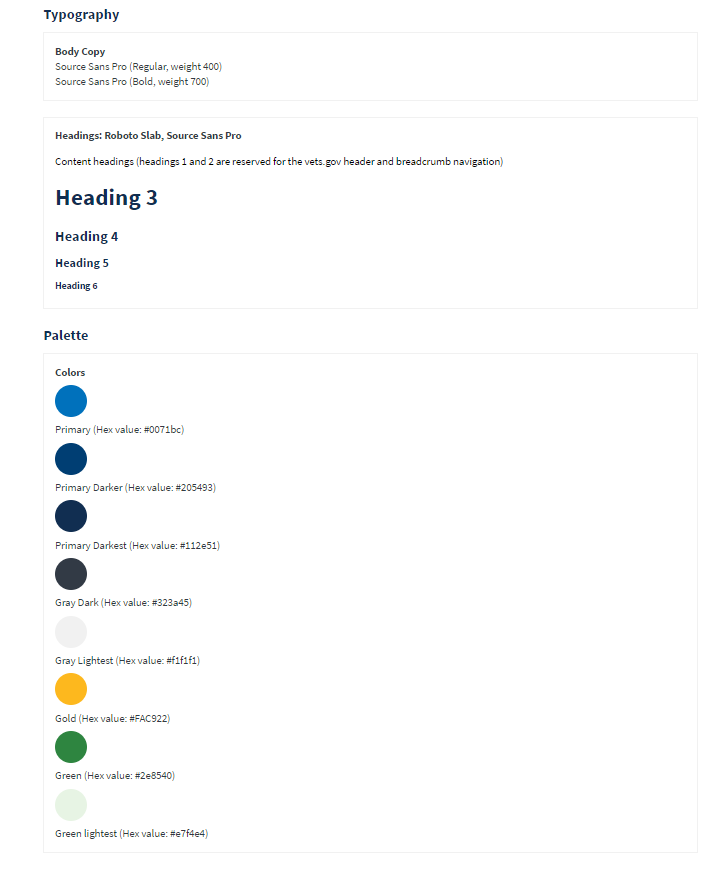
* This document identifies the standard layout and design to be used on each webpage developed for the TAS portal which will be based upon the U.S. Web Design Standards (USWDS) and addresses the above user stories.
* This reference document defines USWDS typography, grid, UI components, headers etc.



* This reference document defines more clearly USWDS typography lists, form controls, buttons, side navigation etc.

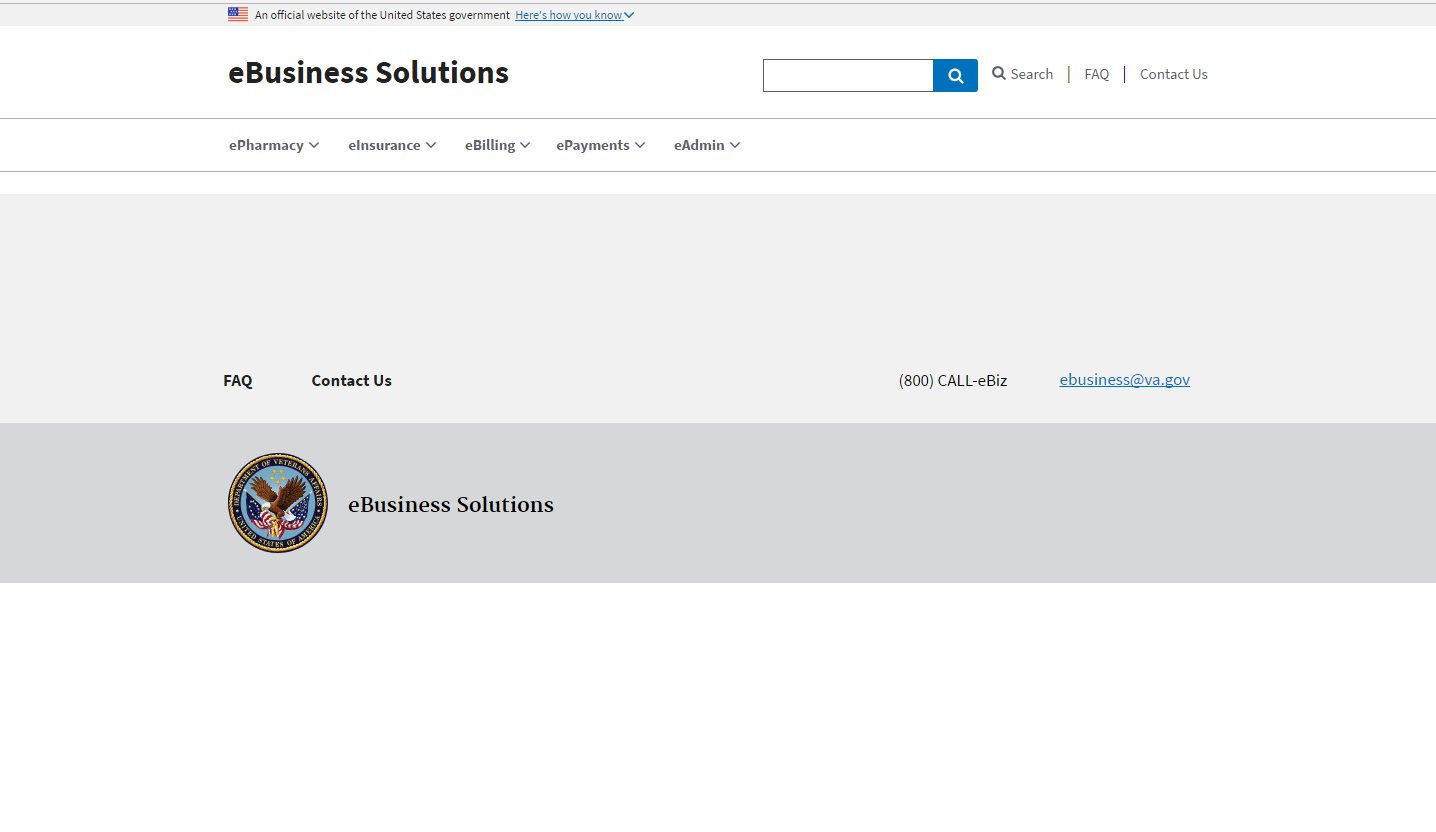


* ``An example of design using USDWS could include:



## **For user stories US1406 and US1409**:

Example prototype of the TAS portal main page has been preliminarily designed using USWDS and will be used as a design template for each of the platform pages including the MCCF EDI home page and the product landing pages - eBilling, eInsurance, ePayments, ePharmacy, eAdmin.

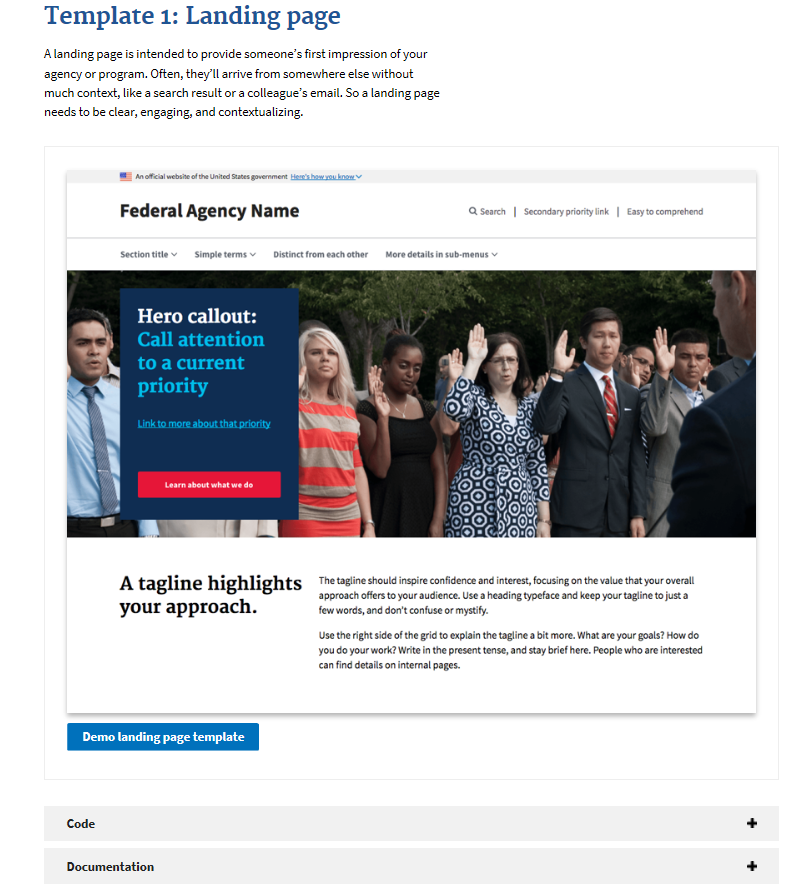


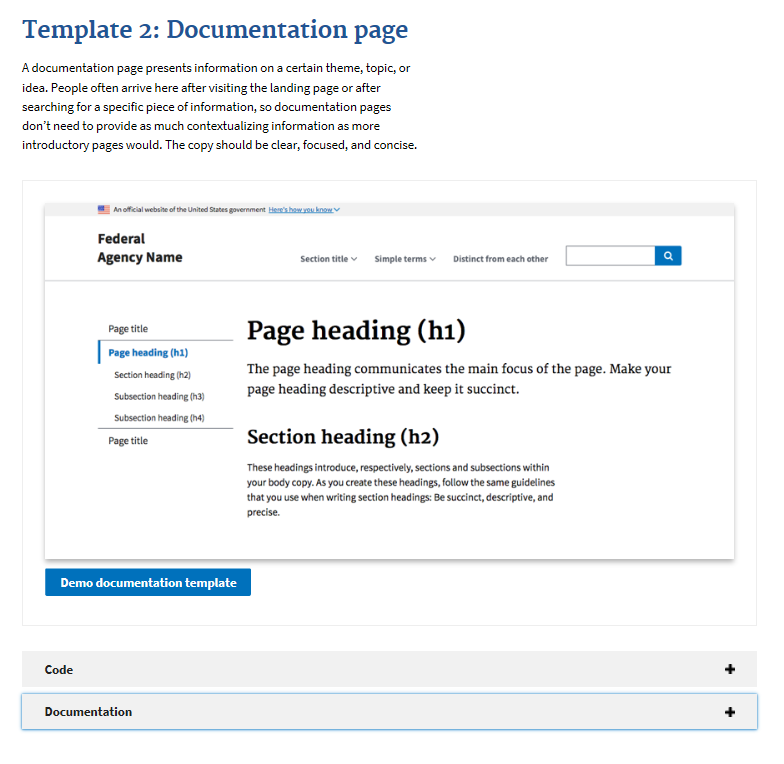
Other high-level requirements such as calendar, news and announcements, alerts / update section, current date / time, breadcrumbs, help and contact list link can be added to the main content section of the page using USWDS components and formatting.

The prototype has FAQ and Contact Us in both the Header and Footer as placeholders.

As a further example of the format intended in the development effort, the following templates can be used:

<https://standards.usa.gov/page-templates/>





# Webpage Error Handling

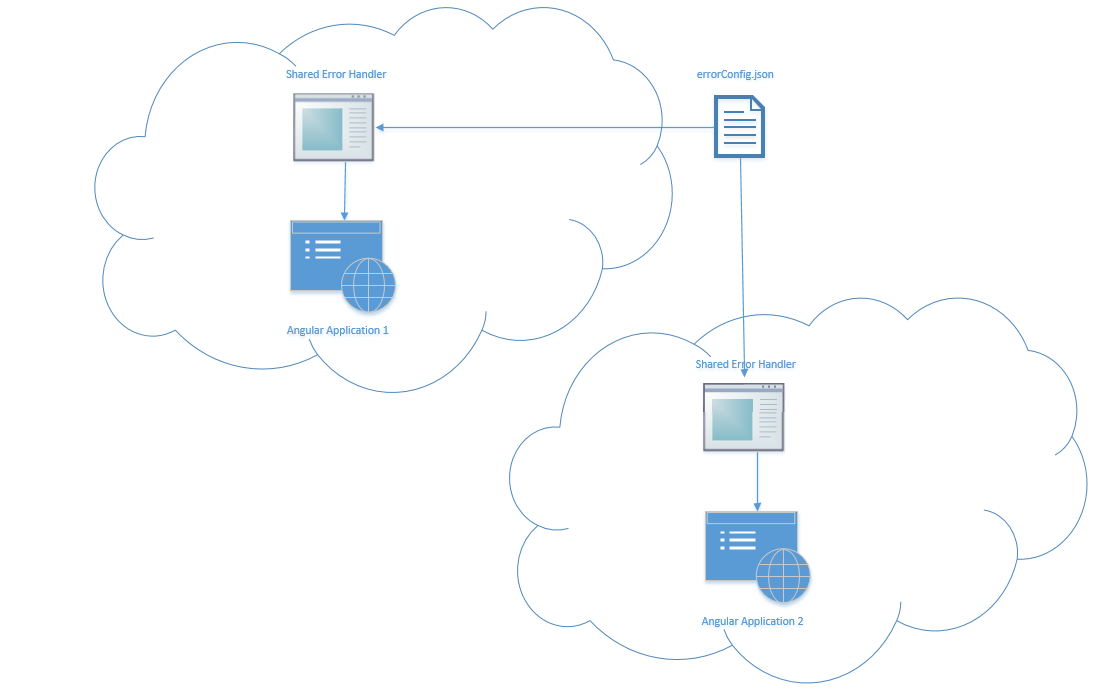
# US1697 - Create a common approach to handle UI errors

# TA3744 - Design UI error handling approach

A shared Angular error handling component (shared component) will be developed which will be used by MCCF TAS Angular applications to handle UI errors. (Each application will include its own instance.)

The shared component could be implemented in a backend service later (the tradeoff will be more backend calls) instead of implementing it in the presentation layer (If page load times become an issue, it might need to be prioritized what is loaded in the UI).

The shared component will use a configuration file which determines how it will handle UI errors and which information to display to the user.



The configuration file will be stored centrally on a file share (location must be defined yet) or in the database. This way it will be accessible by every application that uses the shared component. Editing of the configuration file could be done through a planned content management capability in the future to allow assigning permissions, tracking changes, and approvals.

The configuration file will be in the JSON notation and will have the following structure:

* errors[]
  + errorCode
  + errorMessageUser
  + errorMessageTechnical
* contacts[]
  + application
  + contact
    - name
    - email

The name of the configuration file will be ***errorConfig.json***

Example configuration file:

***{***

***"errors": [***

***{***

***"errorCode": "HTTP 500",***

***"errorMessageUser": "good error message that denotes what is occurring",***

***"errorMessageTechnical": "Internal Server Error"***

***},***

***{***

***"errorCode": "HTTP 501",***

***"errorMessageUser": "good error message that denotes what is occurring",***

***"errorMessageTechnical": "Not Implemented"***

***},***

***{***

***"errorCode": "timeoutVistA",***

***"errorMessageUser": "A timeout occured while accessing VistA. Please try again later.",***

***"errorMessageTechnical": "timeout technical message"***

***},***

***{***

***"errorCode": "default",***

***"errorMessageUser": "An error occured",***

***"errorMessageTechnical": "An error occured - default"***

***}***

***],***

***"contacts": [***

***{***

***"application": "eBilling",***

***"contact": {***

***"name": "First Name LastName",***

***"email": "first.last@va.gov"***

***}***

***},***

***{***

***"application": "eInsurance",***

***"contact": {***

***"name": "First Name LastName",***

***"email": "first.last@va.gov"***

***}***

***}***

***]***

***}***

The error messages will be the same for all applications. The contacts will be specific to each individual application (The shared component will need to determine which application ran into the error and map the correct contact information).

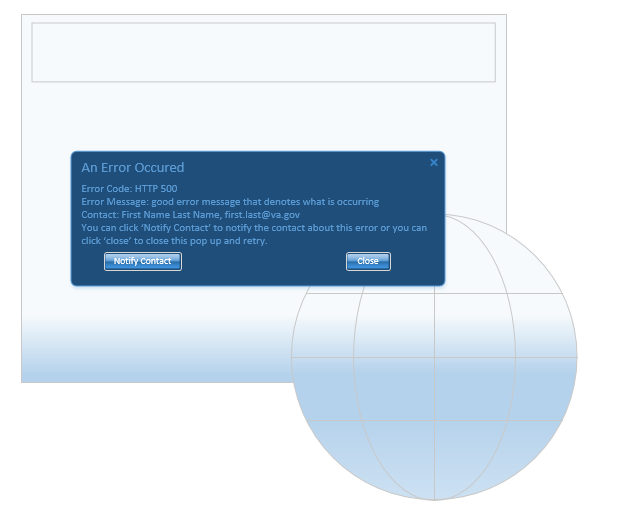
If an application wants to use the shared component, a system admin (yet to be defined) must be notified, so he/she can add the contact information to the errorConfig.json file.

The configuration file must also define a default error element in case the error that occurred cannot be found in the configuration.

The shared component will be loaded the first time the application is called. The shared component will then read the configuration file. If there is a change to the configuration file, the shared component must detect the change and load the new configuration file.

If a UI error occurs the application must catch that error and pass it to its instance of the shared component. The shared component will create a pop up window that will inform the user that an error occurred. The main Angular page will remain untouched and will not display any error related information (i.e. stack trace).

Based on the configuration file, the pop up will contain information regarding the error that has occurred.



When clicking the ‘Notify Contact’ button on the pop up, the shared component will send an email to the contact (errorConfig.json - errorConfig.contact.email). Email messaging will use the SMTP server implemented in the TAS Platform. When clicking the ‘Close’ button, the pop up will close.

The email that will be sent to the contact (errorConfig.json - contacts.contact.email) will contain the following information:

Email subject:

An error occurred in application: *see table below*

Email body:

Error Code: *see table below*

ErrorDescription: *see table below*

Stack Trace: *see table below*

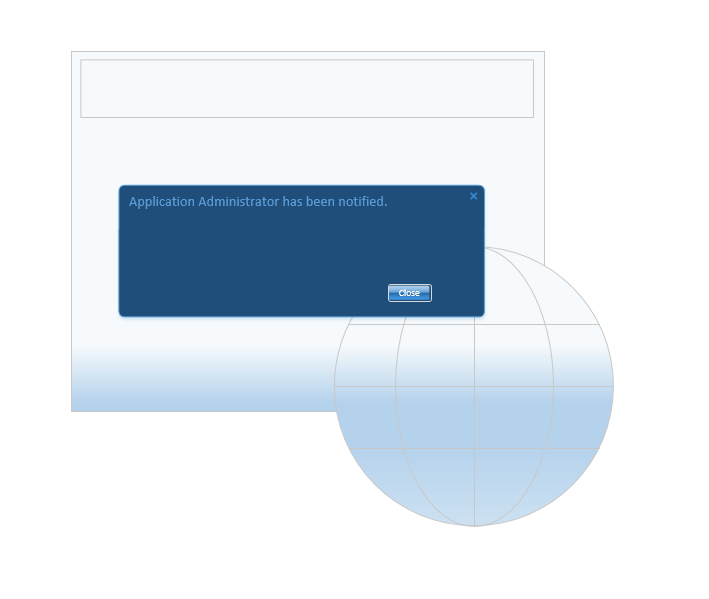
User: *see table below*

Date/Time: *see table below*

|  |  |
| --- | --- |
| Email content | Object |
| An error occurred in application | errorConfig.json – contacts.application |
| Error Code | errorConfig.json – errors.errorCode |
| ErrorDescription | errorConfig.json – errors.errorMesssageTechnical |
| Stack Trace | Stack Trace provided by the system |
| User | Reporting User |
| Date/Time | Timestamp when the error occurred |

If the email sending was successful, the text in the pop up will change to:

**Application Administrator has been notified.**

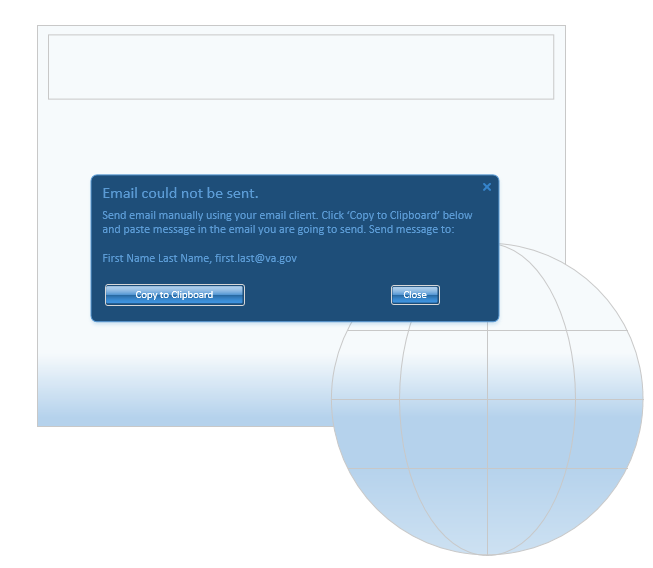


If an error prevents the email from being sent successfully, the pop up text will change to:

**Email could not be sent.**

**Send email manually using your email client. Click ‘Copy to Clipboard’ below and paste message in the email you are going to send. Send message to:**

**First Name Last Name,** [first.last@va.gov](mailto:first.last@va.gov)



The pop up will contain a button ‘Copy to Clipboard’.

Clicking the button will copy the relevant information to the clipboard (same information that would be provided via email – see above).

## **For user stories 1407 and 1408**:

The links to the eRevenue Resource SharePoint Training Site can be added to the MCCF EDI TAS portal home page or the product subpages and can be developed in accordance with the high-level requirements using USWDS components side navigation bar or header menus as described USWDS Sketch PREVIEW.pdf and the USWDS Omnigraffle Preview.pdf files embedded above.

This also applies to the Insurance Capture Buffer (ICB) Web link on the eInsurance product page.

US1693 gives guidance on what happens when a user is about to leave TAS domain so is incorporated into this document.

TAS users need to be notified when they click a link that takes them outside the TAS system.

TAS system is based on Node.js, Angular 2 web system framework.

TAS users can click on links to navigate to websites outside of the TAS system.

TAS users need to be notified when they click a link that takes them outside the TAS website.

The Angular 2 Node.js framework provides a Routing service for handling user navigation using URL addresses. The Router service should catch clicks on URL addresses that are links leading to webpages outside the TAS system. The Router service should present a visual message notifying users that they are leaving the TAS website.

##### User Interfaces

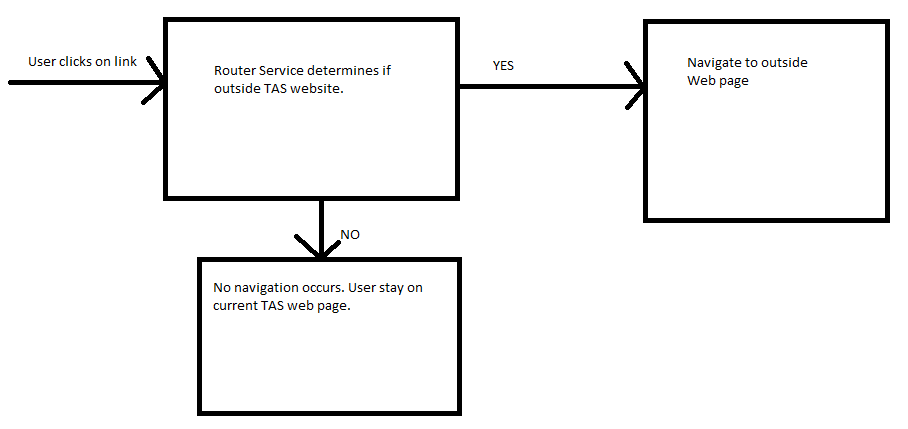
Popup notice window.

**NOTICE**

You are about to leave the TAS website

Cancel

Continue



EXAMPLES

Current government website examples:

