# Structure of the project

The content of the website is structured such that each group/subgroup of conflicts is stored in a markdown (.md) file wherein each conflict is defined by a paneled group of HTML div tags. HTML allows the conflict to be opened and closed, allows styling of the heading and the body, and enables each conflict to have a separate ‘hashable’ link that can be shared.

The important files are listed herein:

* js/customscripts.js – contains common JavaScript functions and is the ideal location to introduce any new functions that are expected to be invoked by the whole project
* css/customstyles.css – houses the common style scripts for whole project (although these styles can be overridden by local changes in the markdown/html files
* pages/mydoc – this directory stores all the conflicts and submenus i.e. all the textual data regarding the conflicts, directories, problem identification and feedback subjects can be found here. In addition, each markdown file which lists the conflicts also stores some local JavaScript functions that enable the ‘link-sharing’ and scroll-to-conflict mechanisms and can be found at the end of the file
* \_includes/topnav.html – HTML file that defines the top navbar which contains the submenus, the search bar, and the Conflict Resolution Index
* \_data/sidebars/mydoc\_sidebar.yml – lists hierarchy of the conflicts within their groups and subgroups
* search.json – lists the links for each and every conflict in the project and acts as the index for the search feature

# Creating a new conflict

As specified previously, each conflict is housed in a markdown file and wrapped in HTML to allow for styling and JavaScript functions.

If you wish to create a new conflict in an existing page, follow the below instructions:

1. Use the following template to write the conflict and note to change the text wherever you find text in caps surrounded by <>:

<div class="panel panel-default">

<a class="noCrossRef accordion-toggle" data-toggle="collapse" data-parent="#accordion" href="#<INCLUDE-HASH-AS-SINGLE-STRING>" aria-expanded="false">

<div class="panel-heading" style="background-color:#f5f5f5;">

<h4 class="panel-title">

        <CONFLICT TITLE HERE>

        </h4>

        </div>

    </a>

    <div id="<SAME-AS-HASH-STRING-SPECIFIED-IN-THE-A-TAG>" class="panel-collapse collapse noCrossRef">

    <div class="panel-body">

        <BODY OF CONFLICT HERE>

        </div>

    </div>

</div>

There are 4 instances where you need to change the text here:

1. The hashstring for the link in the anchor (<a>) tag. This needs to be a single string with no spaces although words can be separated by hyphens. The ‘#’ at the beginning must remain and should not be removed.
2. The conflict title in the <h4> header tag
3. The HTML id of the div tag of the body must be the same exact string specified in the anchor tag earlier (i.e. in the first step)
4. The body of the conflict in the div tag – feel free to use HTML to style the conflict any way you prefer

Once you have designed the conflict, the next step you’ll have to do is to add the conflict and its link to the search.json file. Use the following template to add it as a block to the search.json file:

{

"title": "<CONFLICT TITLE HERE>",

    "tags": "",

    "keywords": "<COMMA SEPARATED LIST OF KEYWORDS HERE WHICH THE SEARCH FUNCTION WILL USE TO IDENTIFY THE CONFLICT FROM THE USER’S TYPED INPUTS>",

    "url": "<HASH LINK TO THE CONFLICT YOU CREATED – IT IS TYPICALLY THE HTML\_FILE#HASHTAG-OF-CONFLICT>",

    "summary": ""

},

1. If you are creating a new conflict altogether (i.e., creating a new markdown file), then you identify the directory in which you want to add the markdown file (based on the group of the conflict) and begin the markdown file’s content with the following:

---

title: <TITLE OF CONFLICT GROUP/SUBGROUP>

permalink: <NAME-OF-MARKDOWN-FILE-WITHOUT-SPACES>.html

sidebar: mydoc\_sidebar

keywords: <COMMA SEPARATED LIST OF KEYWORDS>

# summary: "<IF YOU WISH TO PROVIDE A SUMMARY UNDER THE TITLE, YOU CAN UNCOMMENT THIS AND PROVIDE IT HERE>"

toc: false

folder: mydoc

---

Once you have added the above content to the start of the markdown file, you can proceed with creating the individual conflicts using the instructions outlined in the previous section (i.e., draft the individual conflict and add its title and link to the search.json file to enable the search feature.

Finally, you register the conflict group/subgroup by listing it in the ‘\_data/sidebars/mydoc\_sidebar.yml’ which serves as the Table of Contents for the entire conflict index. This is a ‘.yml’ file wherein the indentation of the entries defines the hierarchy.

# Sub-Menus

The sub-menus – Problem Identification, Directory, and Feedback – are listed under the ‘\_data/topnav.yml’ file. It follows the same .yml structure as before wherein each indent represents an underlying hierarchy. The content of the Problem Identification, Directory and Feedback pages are located in the ‘pages/problem-identification/’, ‘pages/mydoc/directory/’, and ‘pages/feedback/’ directories respectively.

# Feedback

The feedback form is powered by Google Forms which is embedded into the HTML of the page. You can find it in the following path: ‘pages/feedback/feedback.md’. To replace the form with a new one, simply copy the embedded ‘<iframe>…</iframe>’ tag provided by Google Forms and paste it directly in the ‘feedback.md’ file replacing the old link. Adjust the height and width of the form to suit the style of the page as desired.

# Running the website locally

The website is built using Jekyll which is a Ruby Gem (i.e., a Rust library). To get an instance of the site running locally, you will need to begin by installing Ruby. Here are the instructions to do so:

* If you are using a Windows machine, follow the instructions on [this page](https://web.mit.edu/rust-lang_v1.25/arch/amd64_ubuntu1404/share/doc/rust/html/book/second-edition/ch01-01-installation.html). You may additionally need to install Visual Studio C++ Build tools if it’s not already installed. The Rust installation will walk you through the steps of installing it and contains helpful troubleshooting references in case the installation runs into any issues.
* If you are using a Mac, open a terminal and run this command: 

Once you have installed Rust, you will need to install Jekyll, the instructions for which can be found [here](https://jekyllrb.com/docs/installation/).

With the Ruby and Jekyll setup now complete, you will need to clone the GitHub repository that contains the code for the website from [here](https://github.gatech.edu/ehariharan3/apollo.github.io). (If you don’t have Git in your system, please install it by following [this link](https://github.com/git-guides/install-git). There are also some helpful introductory Git commands [here](https://www.atlassian.com/git/glossary) that will come in handy.)

Once you have cloned the repository in your local system, open a terminal (Mac)/Ruby Command Prompt (Windows) to that directory and run the following command:   


And you’re all done with the setup!

**NOTE:** If you only wish to make content updates, you do not need to setup the website locally. The website is hosted on GitHub Pages, and you can sidestep the time-consuming installation process by making updates to the files directly in GitHub and testing them on GitHub pages directly.