National University of Sciences & Technology 

School of Electrical Engineering and Computer Science

Department of Computing

**CS-213: Advanced Programming (3+1): BSCS-7AB**

**ASSIGNMENT 1**

# SUBMITTED TO: Dr. Sidra Sultana

# SUBMITTED BY: Amna Muqeem (216259)

# Class (Section): BSCS 7B

# Date of Submission: 16th Sept, 2019

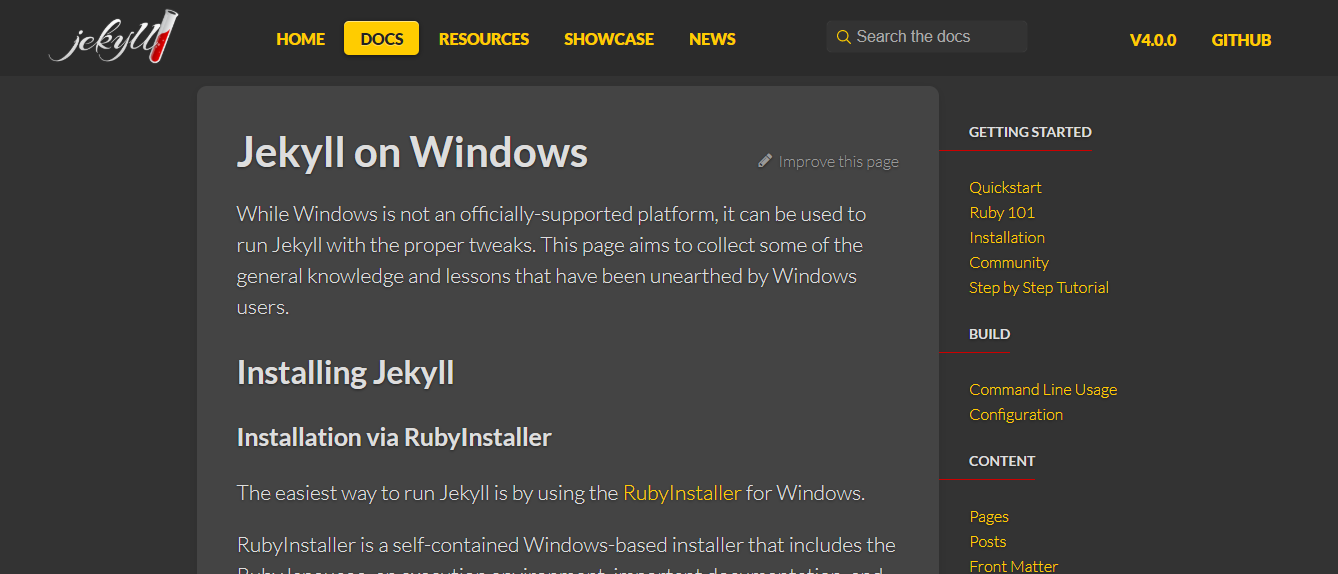
## Configure any open source software (OSS)

## OSS Chosen: Jekyll 4.0.0

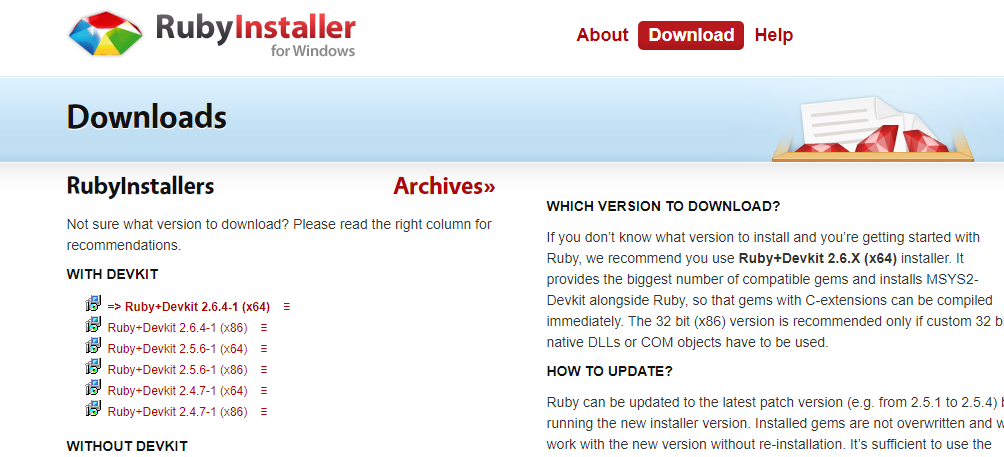
1. **Downloading**

For the first task I chose Jekyll 4.0.0 which is an open source static website generating software written in Ruby by Tom Preston-Werner, GitHub's co-founder. It is often used for blog sites in accordance with GitHub.

Jekyll isn’t directly available for download through Windows due to Windows not being an officially supported platform. So, in order to install it, the website states the use of RubyInstaller.

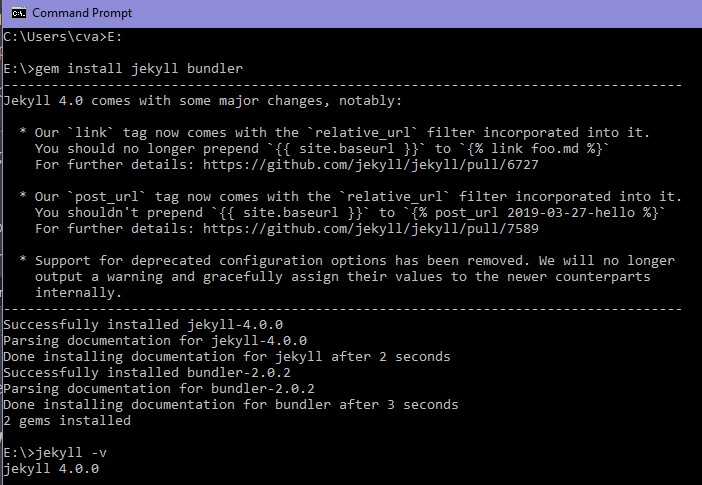


I downloaded the recommended Ruby+Devkit 2.6.X (x64) in order to further install Jekyll.



1. **Installation**

Upon completing the ruby setup process, a Command Prompt Window opened which was further used to install various packages. I followed the guidelines given on Jekyll’s websites and ended up installing all the required gems for the proper functioning of the software.

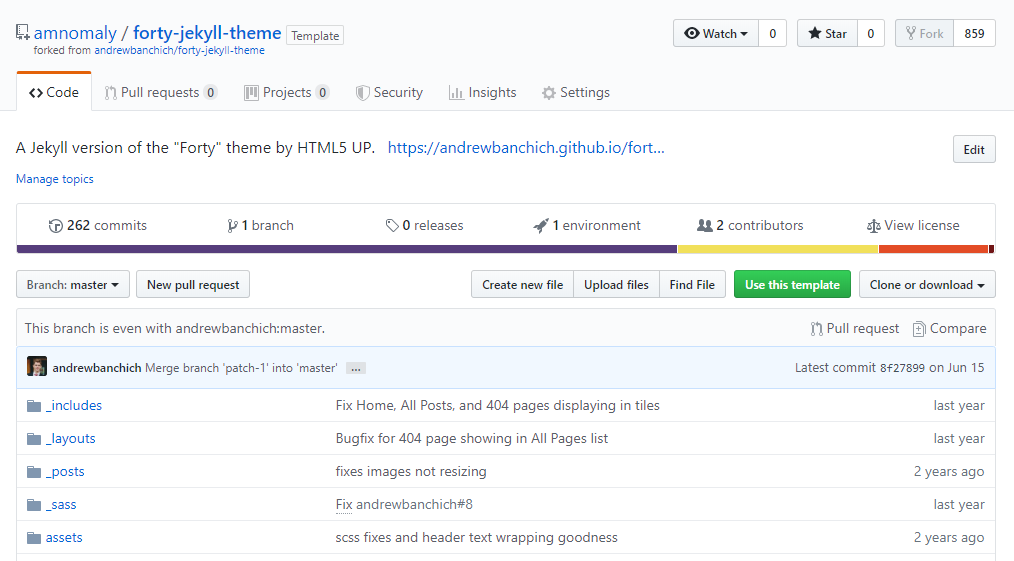


After getting the confirmation that Jekyll was finally installed, I was good to go.

1. **Explore it with user’s perspective**
2. **GitHub and Website Setup - Admin’s Perspective**

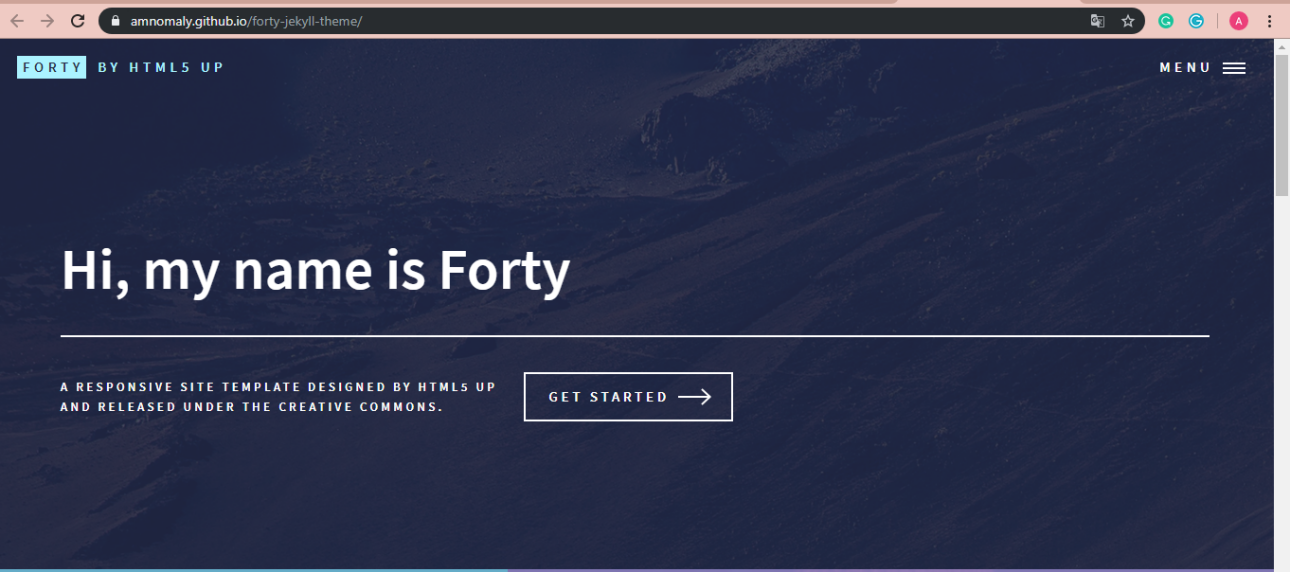
Now, Jekyll won’t work as it is. It needs a host or a server to actually hold all the files that one may have in order to execute a website. In this case, the host that works in accordance with Jekyll is GitHub. So, in order to get my website up and running, I needed to have a GitHub account.

After creating a GitHub account (that I previously didn’t own), I made a repository. Then I looked up a theme using the keyword “jekyll” on GitHub and forked it (i.e. transferred the content files of the theme) from the owner’s repository to mine.



The interface is quite easy to navigate through from an admin’s perspective. All the code files are displayed under a separate section while the relevant settings are found under another section.

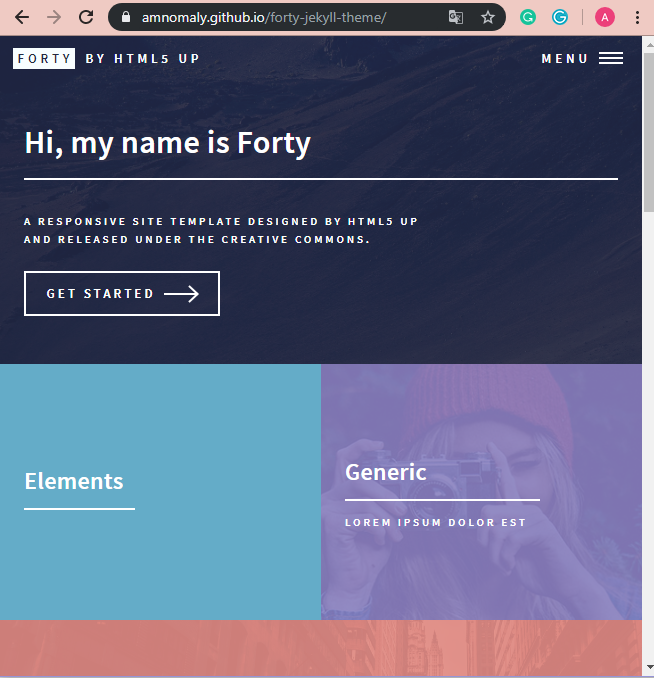
After this, I navigated to the settings section, where I waited for the generation of my link. Once it was generated, I could finally access my website at: <https://amnomaly.github.io/forty-jekyll-theme/>



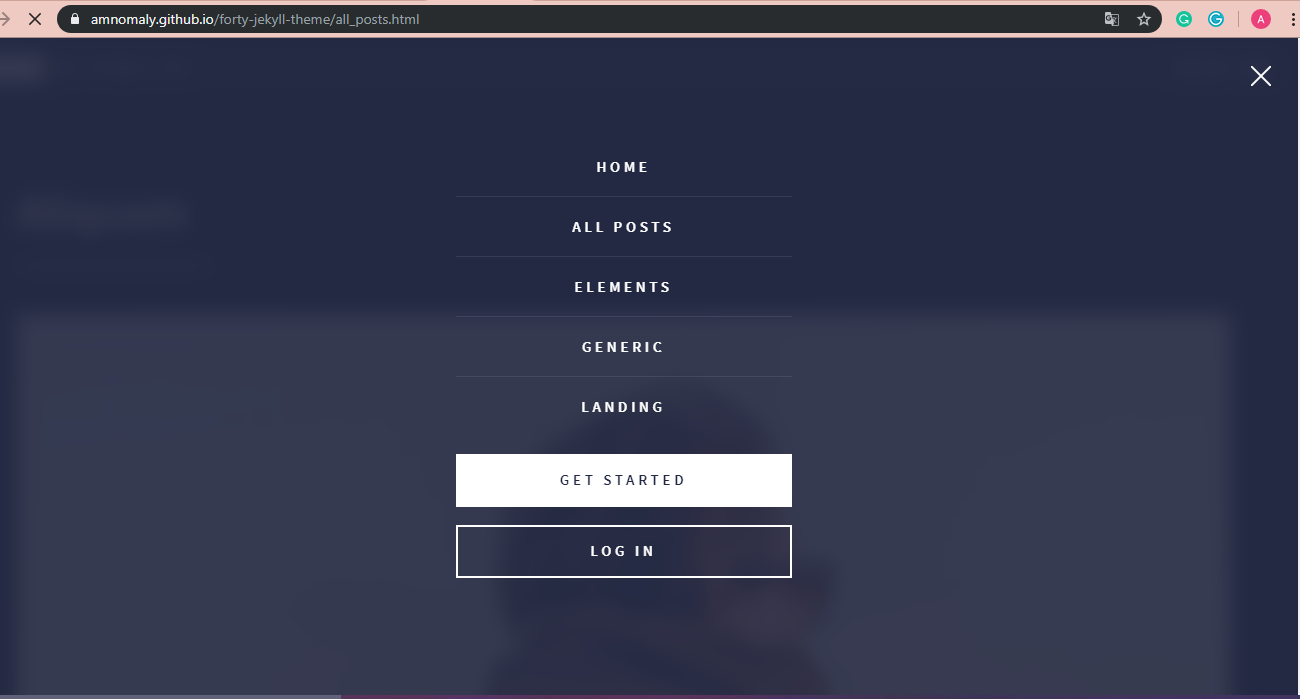
The theme I used belongs to Andrew Banchich and is called “Forty”.

1. **Navigation - User’s Perspective**

From a user’s perspective, once they land on the homepage of the website, they’re shown multiple clickable banners that lead to various other pages of the website. The website is responsive for many devices and screens.



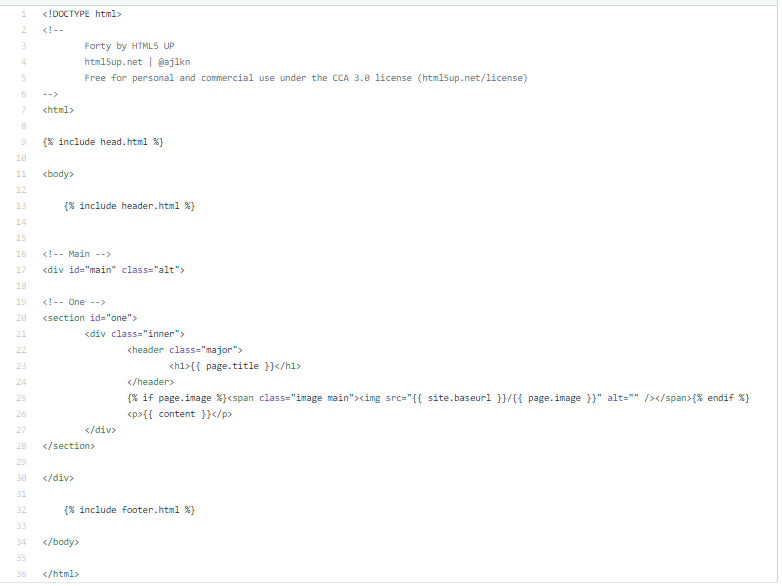
It has a main menu that holds multiple pages, making it very easy for the user to navigate through the site.



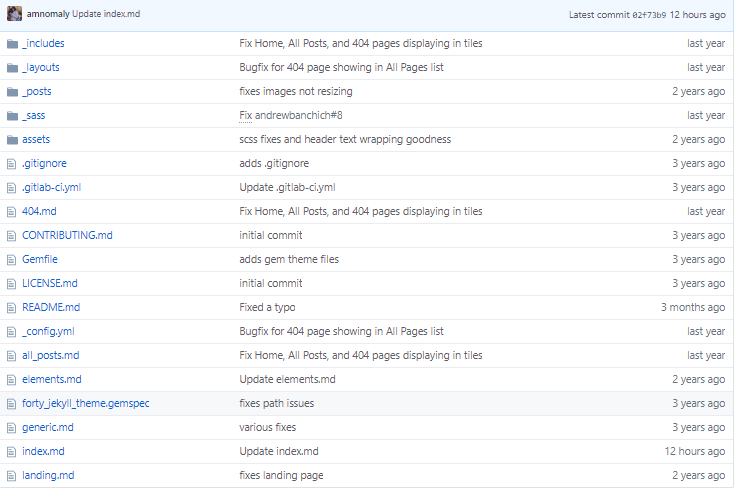
The website currently shown in the above screenshot is in its basic form. With only a couple of modifications in the code, it can be adjusted to the look and feel the admin may prefer.

1. **Understand the database and coding structure**

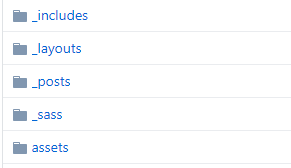
Jekyll is a static website generating software, thus it does not deal with databases. We may explore its code, though, in order to understand the working of the website.



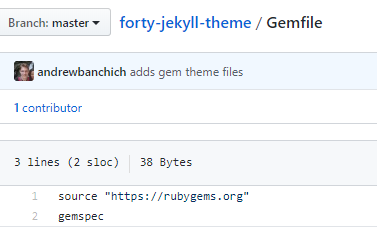
The basic code and files work in a way such that the header and footer, home, etc. files are separate. On each page, one can easily include them by a simple line: “{% include head.html %}”. This allows for simpler and easy to read code instead of large chunks that are difficult to navigate through.



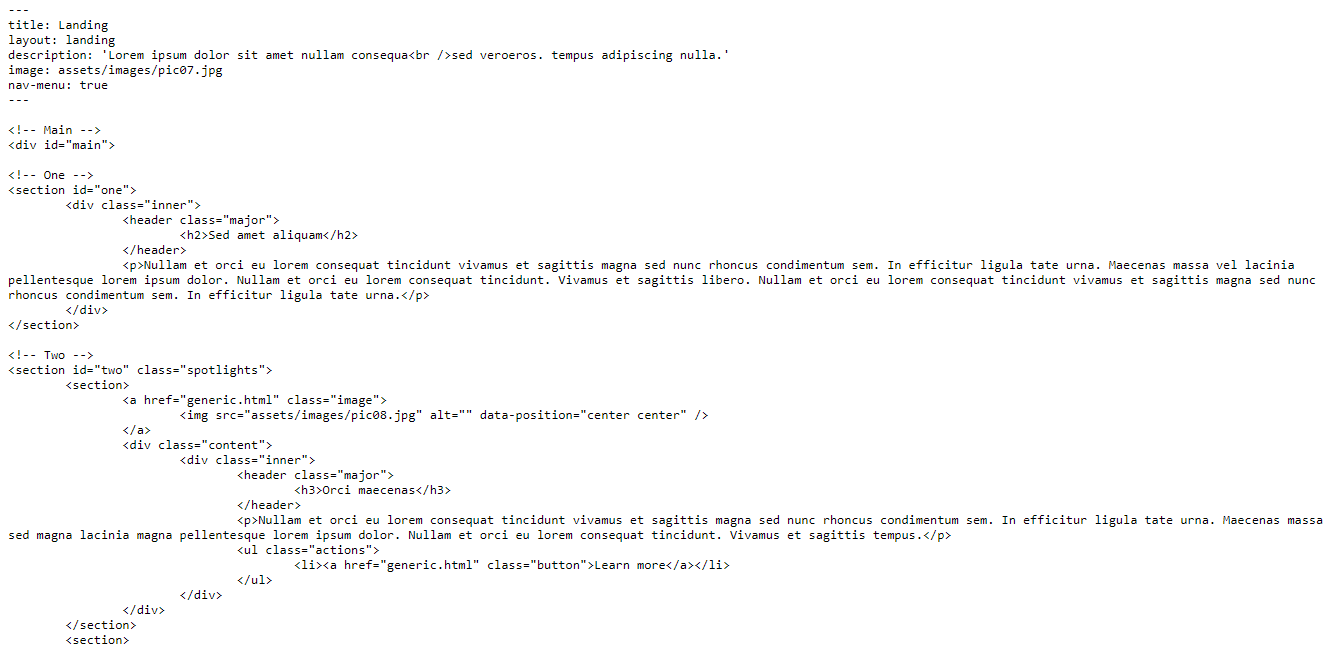
This was the list of files that I forked in order to use as my own and create the website in my own repository.



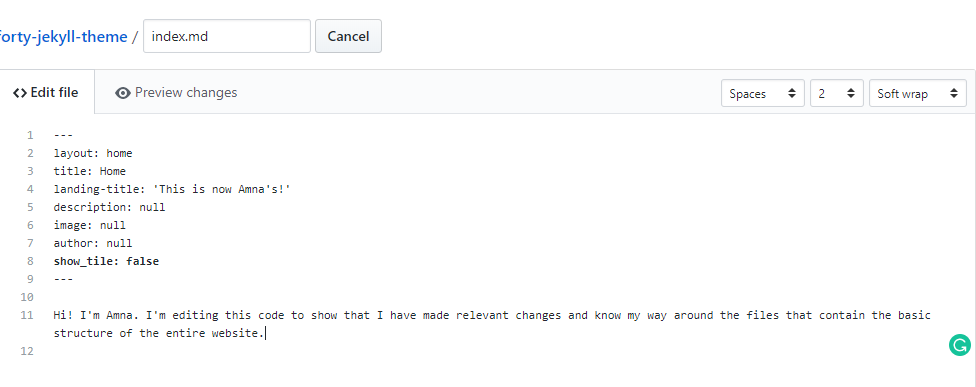
These folders consist of all the base html files for the website that can be easily manipulated to suit one’s needs.



The Gemfile is a familiar name. It is a file that is created for Ruby programs in order to describe the Gem dependencies. A gem is a collection of Ruby code that is called here from rubygems.org for the proper functioning of the website.

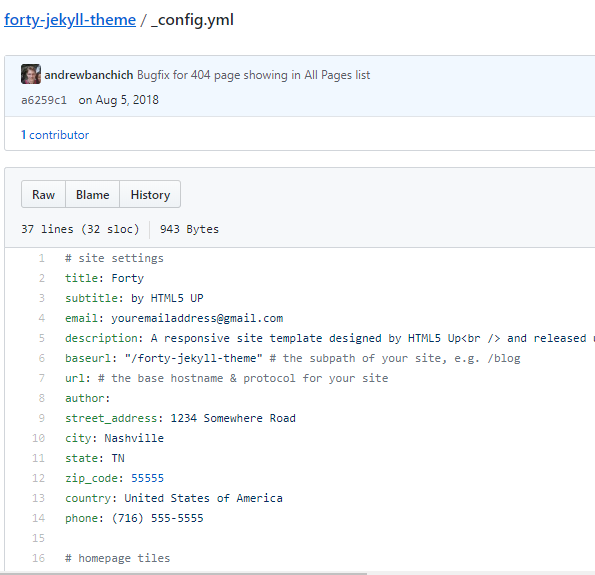


The file landing.md consists of the code for the landing page with various divs, sections and images. GitHub shows this code in a more friendly way so that it can be easily manipulated while keeping the theme intact.





I manipulated the index.php file to change its content from Lorem ipsum to my own content.



\_config.yml is also an important file that I used to deal with the 404 error for the website. It is a file that lets the person who downloaded it make changes to the plugins without having to edit the source code. Usually by manipulating a small factor in the said file, one can easily get rid of errors.

Initially, I couldn’t get the link for the website. I looked the problem up and the solution was fairly simple. I had to edit the \_config.yml file in order to get the link as soon as possible. This issue usually arises when your url clashes with another url, thus you have to change the base url of your site through the mentioned config file.

Another issue that I got was that even when I clicked on the link that I was provided with, I was still getting a 404 error. I easily got rid of this by clearing the cache and browser history.

1. **Use some online available plugins or modules**

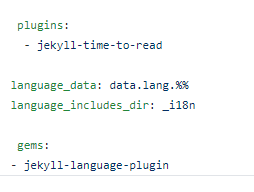
Jekyll offers multiple services in the form of plugins like: [Generators](https://jekyllrb.com/docs/plugins/generators/), [Converters](https://jekyllrb.com/docs/plugins/converters/), [Commands](https://jekyllrb.com/docs/plugins/commands/), [Tags](https://jekyllrb.com/docs/plugins/tags), [Filters](https://jekyllrb.com/docs/plugins/filters/), and [Hooks](https://jekyllrb.com/docs/plugins/hooks/).

For this website, I tried to implement [jekyll-youtube](https://github.com/dommmel/jekyll-youtube), readingtime.rb, jekyll-language-plugin, and jekyll-time-to-read. The first three were .rb files that were coded in ruby, while the last one is a gem file.

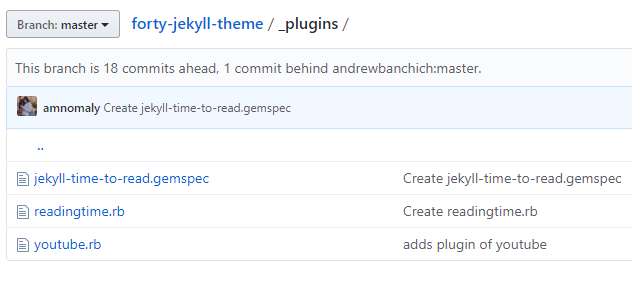
The syntax for all three of them included “{{ content | reading\_time\_as\_i }}” where you had to write the name of the plugin beside the content where it is supposed to appear in the web page.



The \_config.yml file also had to go through various changes for the plugins to work properly.



I had made a completely new directory in the master directory, by the name of \_plugins, where I stored all my plugin files. Each file had its own code that was used to implement the plugin.



But eventually, even after making all these changes, the plugins couldn’t run properly. This was due to the fact that GitHub disables custom plugins for all the sites and pages that are generated using the --safe option. This is due to security reasons.

Regardless of that, I did see a couple of websites that had implemented these plugins and learned how those websites used those plugins to get the structural changes that they required.

1. **Conclusion**

Navigating through Jekyll taught me how open source software operate and how they are implemented by users. I also gained familiarity with GitHub and its templates, themes and plugins. I got to understand the code and the arrangement of files in Jekyll. Overall, this was a great learning experience that I can use for exploring more OSS in the near future.