```
 ____                    ____  _
/ ___|  _____  ___   _  |  _ \| |__   __ _ _ __ _ __ ___   __ _  ___ _   _
\___ \ / _ \ \/ / | | | | |_) | '_ \ / _` | '__| '_ ` _ \ / _` |/ __| | | |
 ___) |  __/>  <| |_| | |  __/| | | | (_| | |  | | | | | | (_| | (__| |_| |
|____/ \___/_/\_\\__, | |_|   |_| |_|\__,_|_|  |_| |_| |_|\__,_|\___|\__, |
                 |___/                                               |___/
 ____        _        _
|  _ \  __ _| |_ __ _| |__   __ _ ___  ___
| | | |/ _` | __/ _` | '_ \ / _` / __|/ _ \
| |_| | (_| | || (_| | |_) | (_| \__ \  __/
|____/ \__,_|\__\__,_|_.__/ \__,_|___/\___|
```

SE-Project

Flask Pharmacy stock manager website with an admin system.

Further Work

Frontend:

The front end uses simple HTML generated using something called jinja. If you look in the templates folder you'll find the templates used for the different pages, but they use jinja so they're slightly different from normal HTML.

Understanding jinja syntax

If you look at layout.html, it*looks* like its normal HTML, but you'll notice some weird syntax like there's some stuff in curly braces. This is one of the areas where jinja is different from HTML. Those curly braces can be used to run python code in the template itself before you create your HTML page. So in layout.html, one of the things I'm using python for is to decide what the title should be.

Now, what is layout.html? I'm using it as a template for all my pages. If you go through it, you'll notice that layout.html has a head tag, a body tag and a navbar tag but not much else. And the other pages, home.html, about.html etc don't have a head or body tag. This is because all our pages will be having some same elements, like the navbar, some other general HTML tags like meta tags and the CSS rules. So jinja allows us to create a general template and then extend it in each of our pages. Layout.html is that template.

Now how do we make a page from this template? In the layout.html template, you'll find a div with the class 'container'. This div contains some python code that you can ignore, but at the end, you'll see:

```
{% block content %}   {% endblock %}
```

This provides a spot in the layout template where you can insert code and in this above example that spot is named 'content'

If you open one of the pages(for example, about.html), you'll notice that the page starts with

```
{%extends 'layout.html'%}
```

As you would expect this is basically telling jinja that "my final HTML page that you generate should have all the content from the layout.html file". After the extends block there is a block saying block content. All of the content of that particular page is placed inside the {%block content%} and {%endblock content%} tags. This tells jinja to put the contents of this page into the layout template in the spot where the layout template has the block content tag as mentioned above.

Another weird thing about jinja is how you link a css stylesheet. Usually you would put the following tag in your html page

```
  <link rel="stylesheet" type="text/css" href="static/main.css">
```

But in jinja you have to use

```
  <link rel="stylesheet" type="text/css" href="{{ url_for('static', filename='main.css')
```

and all the css and javascript files you use need to be put in the static folder.

I have put my main.css file there(you can add additional CSS rules here or in another file but if it's in another file you have to add the link tag to that file in the layout folder).

Styling

I am really terrible at styling. If you want to see just how terrible I am at styling, go to the layout page and remove any lines that contain the word bootstrap in them and reload your page. The ugliness you now see is what I styled. I used bootstrap to do most of the styling for me. Bootstrap is a CSS stylesheet that is predefined for you to use. If you looked through my HTML, you would have noticed that almost all my HTML tags have multiple classes, like in home.html, my h1 tag has been given both the display-1 and title class. The display-1 class is predefined by bootstrap and so it is styled according to that and the title class was something I gave it so that I could add a CSS rule to it in the main.css file if I wanted to. The best example of the power of bootstrap is the navbar. I just wrote a list of my links(which looks really ugly. if you remove the lines of layout.html and reload the page you'll see what I'm talking about) and put the appropriate bootstrap class for the links and I got the B-E-A-U-titful navbar that you see on the page.

If you want to use bootstrap on an HTML item, just google "Bootstrap paragraph" or "Boostrap table" and go to the bootstrap website, look through what the different classes look like and copy-paste the class you like to your HTML page.

If you want to put a custom style on an HTML item just put a unique class on it and define its style in the main.css file in the static folder.

Note

I have downloaded and linked the bootstrap CSS, bootstrap javascript and the jquery files so that you don't need an internet connection to download those libraries every time to see what your webpage looks like.

The Forms

Flask doesn't use normal HTML forms for input, it uses a python object to make using forms easier in the backend. If you look at any page using a form, like the login page, the registration page, or even the search bar, there's a lot of complicated python and jinja syntax that you don't really need to understand. To style them just give the

<

div> tag that they are in a class and put whatever style you want.

If you want to change the text that appears above a particular input field in a form then you need to change the label of that form element in the forms.py file (Mayank should know how to do this by now)

The big text at the top of a form can be changed by changing the text in the legend tag.

Short note on the tables

I made a template on tables so that I don't have to update each page every time I want to make a change to the table. The template is in the table.html file and to include it in a page I just add the

```
{% include 'table.html %}
```

command wherever in the HTML page I want the table to appear. If you want to style the table you can style it in the table.html file, and if you want to style it only for one particular page you can copy-paste the code in the table.html file into that page(after removing the extends and block content lines) and style it there.

Note

A cool feature that I added was that I used a javascript library that I found to sort tables dynamically in the browser. So if you click on the header of a column in the table the table will sort according to that table. Now for this to work the table has to have the sortable class, So don't remove it if you want this feature.

Todo

The delete functionality is still remaining to be implemented. Haven't learned how yet