Capstone: Operating Instructions

Austin Cooperrider

Trident University International

CSC 425: BSCS Integrated Project

Mr. Michael Greco

11 Oct 22

# How to start

After downloading all files, open a windows command prompt and change directory to the folder that contains ‘main.py’. From there run the following command ‘python main.py’. This will launch the program and provide you the link to access the website. To access the website, just enter the link provided by the application into a web browser. The default link should be <http://127.0.0.1:5000> . Once you gain access to the website, all remaining functions will be through the web user interface.

Accounts already built in the database

a.admin : adminadmin

u.user : useruser

d.disabled : disabled

## Key requirements

The most current version of python is required to run the program. If you do not have a valid version, it can be downloaded via their website <https://www.python.org/downloads/>

Additionally, you can run “python pip install -r requirements.txt” in the main directory to install all required modules.

Due to not being able to upload as a zip file and the dropbox not allowing folders, you will need to move the files into the following file paths displayed below. Alternatively, there is a script setup.txt that will create the folders and move the files for you. You can either copy and paste the commands into command prompt or turn it into a batch script.

**Working Folder**

main.py

Operation.docx

requirements.txt

**website (Key note: do not capitalize)**

auth.py

database.db

models.py

views.py

\_\_init\_\_.py

**static**

**templates**

adduser.html

admin.html

base.html

chng.html

home.html

login.html

ticket.html

userpage.html

## Key Application Functions

The first function will be seen in the login capability and the fact the program can identify what type of user is logged in. a.admin and u.user are admin and standard user accounts. The user account does not have easy links to some pages because they do not have elevated privileges. Those pages have further safeguards to ensure that even if the user gets to those pages they can not commit any edits. The most obvious item is a user can not close an open ticket, where an admin can.  
 The most important program function is the ticket creation and editing. The home page displays all of the tickets with some details about them. The buttons with the ticket ID are links to their specific html pages which preloads all data in the ticket page. The associated back end functions for the ticket page identifies if you are editing a ticket or creating a new one. This prevents creating multiple tickets for the same problem. Entries for the tickets will be displayed and saved on this page, thus allowing seamless updates.

The last key function is the adding and editing of users to the website. The admin page has links to the add user page along with similar capabilities to the home page and ticket page relationship. So an admin can initiate a new user addition or edit a user status with some safeguards in place.

## Status

**ALPHA submit**

As covered previously, the user account functionality is complete. The handling of elevated user status still needs to be fine tuned. Essentially, any function that we want to limit to only admins needs to have checks embedded. For the majority of those functions, that has not been added due to ensuring overall functionality first.

For the home page, I am currently displaying the tickets as a list item. Future design work will be conducted to display the ticket information in an easier to read format. Additionally, a search function has not yet been implemented on the home page. For the ticket page the key function I need to implement is the creation of note entries in the database and the associated displaying of them. From there, how those entries will be displayed. Additional display item problems is the fact that the dates only display the date and not the time.

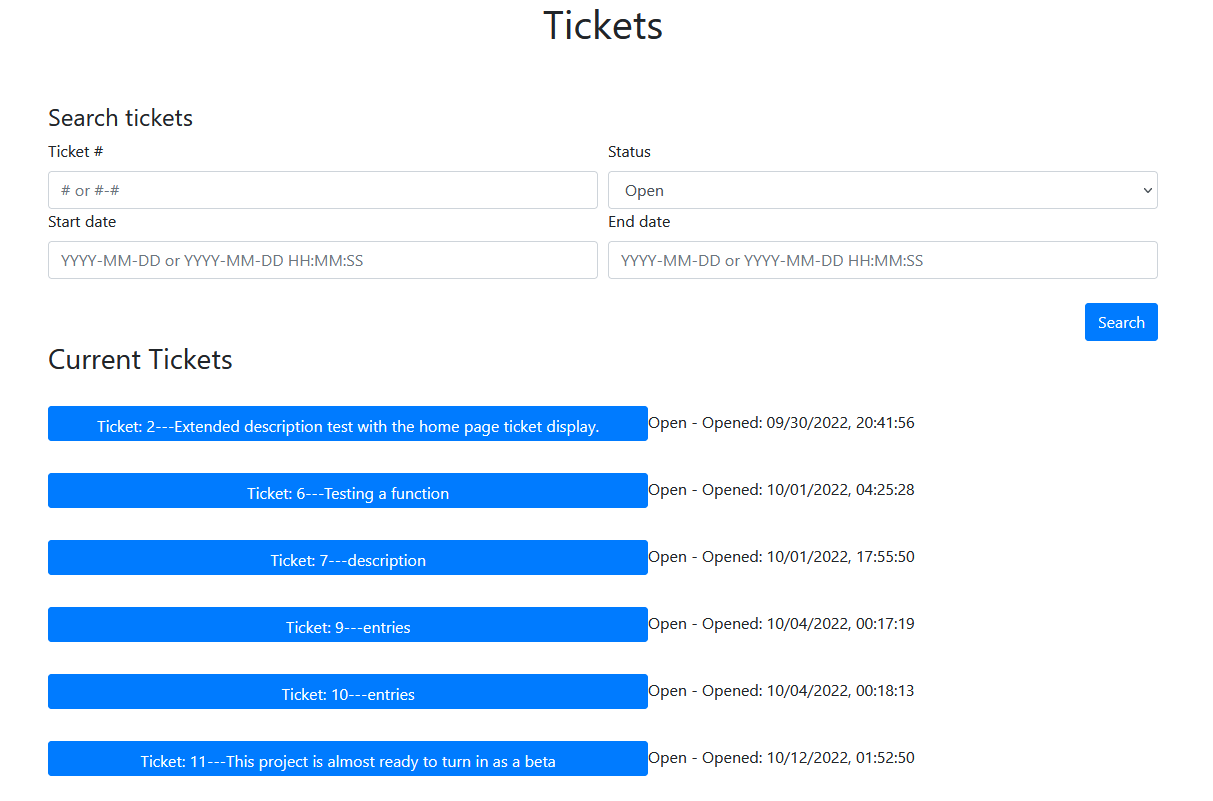
Lastly, the admin pages and their functionality need some improvement. At this time, all they are doing is displaying the current info. The next step would be to edit the pages to actually display similarly to the ticket page and then allow database edits from there.

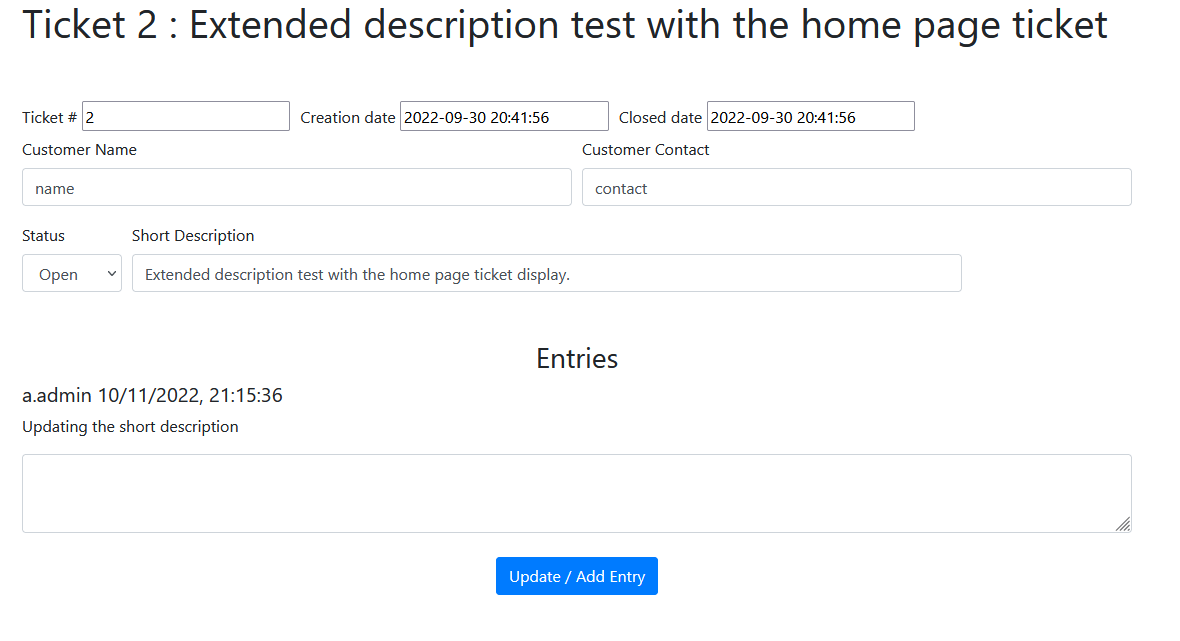
Overall, all of the remaining items are reasonably doable and just will take time to go through, code every item, and debug any errors.

**BETA submit**

All user functionality has been completed along with the establishment of a disabled user account status. This is to ensure no breaking of entries in removing a user. The majority of user privileges checks have been established across the website, but edge cases should be identified and removed during the testing phase.

The home page search functionality has been completed. You can now search based off of the ticket number or range of ticket numbers, open or closed status, and a range of date times. Additionally, the display of tickets has been streamlined along with establishing the ticket number and short description as the button to click to review the ticket. Entries have also been implemented along with a easy to read format differentiating between different edits. The previously mentioned display errors were fixed after identifying that when providing a python variable to HTML, ensure to put them in quotations so html does not confuse a long string as multiple tags.



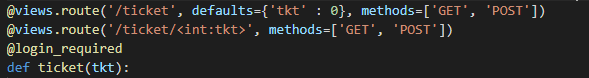


The admin page has been mirrored to match similar functionality as the home and ticket page. Search functionality is complete along with display capabilities. Additional search capabilities had to be added for different types of accounts and most common account groups.

All remaining work that needs to be done is quality control of user input and account verification across the website. Junk entries will be input to see how it breaks the search functions. As mentioned before, meticulous checking of admin status at key points will also be conducted and corrected if need be.

## Challenges

The biggest challenge was how to preload the information for specific tickets. Not wanting to duplicate code, I instead was able to employ subroutes. Whenever a button on the home page for a particular ticket was clicked, it directed to the tickets/<ticket number> page. Utilizing a subroute along with the default route on the view.py tickets page provided a way to pass the ticket number as a variable. From there, a simple IF statement allows the display of information associated with a particular ticket or presents the blank defaults. This also makes it a feature that if you know the ticket number you need to edit, you can just hand jam it into the url.





The next challenge was the implementation of the search functions. In combination with how a query of tickets was preloaded for the whole page, the goal was only one query to run at a time. Instead of having a nasty series of nested if statements with differing permutations of a query function, I was able to identify a means of building a variable to append individual query terms to. This drastically shortened the amount of code and made it far easier to understand.

