Gist of Midder

Midder is a text-based, non-online Q&A program, similar to a forum. By first entering a user ID (optional), users will then be shown:

- Total Question posts
- Total Answer posts
- Average Question score
- Average Answer score
- Total votes

After being shown this summary of the user's statistics, they will be able to:

- Post a Question
- Search for Questions
- Provide an answer to a question
- Vote a selected question or answer

Through simple commands in the console, the user can also choose to exit from the "navigation" screen by typing 'exit'. More details are available below.

Simple Navigation

Midder uses a CLI (command line interface) to interact with the user. First, to use Midder, users, if an account exists, can "log in" to see a summary of their statistics on Midder, however it is not required . Once logged in, the user will be taken to a "navigation" page, allowing users to search through existing posts or create their posts. To take "actions" on a post, a user must first search through posts and select a post. Once selected, a user will be allowed to either vote on the post, answer the post if it is a question, or list all the answers to the post if it is a question.

Components of Midder

The functionality of Midder can be broken down into two major components and subcategories inside:

- The database which stores the information
    - Deals with loading 3 JSON files into the database
    - Then updates the database as the user proceeds to manipulate data.

- The main function, which deals with "Navigation of Tidder."

- Post operations, which encompass all actions a user can make on a post (including searching posts)
    - Login of the user
    - Making a post
    - Searching a post
    - When the user finds a question, he/she can answer or vote on it

We will take a look at each of these components in detail.

Most In-Depth Look at Midder

Midder uses Python, with the implementation of the pymongo module. The use of MongoDB is to create, store, and modify the database, which holds all the information in Midder. The program uses two files to function, each of which is described by the functionality above. We will go through the program's interface and describe what the program in the background is doing.

There are two files the program uses to run: main.py and Database.py.

We start from the terminal where the user types in:

1. Running only the database as it is:
   python3 main.py (A port number)

The first program to run will be main.py. Main.py is the interface of the program, where most "major" decisions take place. When this runs, it will immediately reference the other file, database.py. The database.py file contains a class named Database_Midder. This class, along with its methods, is the primary interface between Python and MongoDB.

Midder database is initialized from 3 JSON files containing the tables posts, tags and votes. When main.py runs, the first call is made to the method login() to create an account, if needed. Once logged in, or the user chose to enter as a guest, next is where most of Midder's functionality lies: In the navigation screen. Users can decide to make a post or search posts, or if they want, they can exit the program by typing 'exit.'

The main.py file is where most of the methods involving interacting with posts are taking place. The main() function, while navigating, will call other methods for all of the "navigating." Two of the most extensive methods in this class are makepost() and searchpost(). makepost() is a smaller method call, not needing any other class methods to run. search(). However, these two contain much of the user's capabilities, such as posting and searching questions.
Below is a short statement for the functionality of each of the methods in main.py:

- login() - Generates a report of user info if they choose to login in
- makepost() - Determines if the post is a question or an answer post. If it is an answer post, then the parent is the question post the user is replying to.
- search() - shows posts that match with keywords typed by the user.

- postaction() - When the user searches and selects a post, then the postaction method is called to show further operations to the user that can be performed on that post.
- listanswers() - List all the answers to a selected post (if that post is a question).
- vote() - Enables user to vote on a post.
- avg() - Returns average
- goodsum() - Returns sum of items in a list.

Testing Strategies

Many bugs originally arose in the first iteration of the program. To make the program fool-proof, testing was done by intentionally trying to break the program, using the knowledge of how it was created. With each new change, added method or function, or any change to the user interface, the program was tested by only testing the feature that was currently being worked on. The testing was strictly restricted to ensuring all the functionality that was described in the specification was met.

As an example, there is a method named vote(). It is the method called when the user wants to vote on a post. The specification document tells us that a user can vote only once. When we created the voting method, users were allowed to vote infinitely, which was discovered by intentionally voting twice or multiple times.

We tested the new methods as we progressed through the program. We tried out different branches of possible results to make sure there are no gaps. With each little change or addition to the program, that addition or change was immediately tested, along with testing all the previous functions and functionality, to ensure none of the new changes broke any of the old code.

Group Work Strategies

Maksimus and Ferdous worked together to create Midder. A general breakdown of the work each partner contributed is found below:

Maksimus:

- Main Function:
- Making post
- Searching posts
- Selecting posts
- Voting on posts
- Answering posts
- Bug fixing

Ferdous:

- Bug fixing
- Database class
- Importing the JSON files

- Handled connectivity with the database

Email was the main method of coordination between us. We first decided who would work on what part of the project, attempting to divide the workload evenly. Next, over the course of two weeks, we emailed back and forth, exchanging code, detailing changes we have made and asking each other to test the program back and forth. Thus, how the creation of Midder came to be.