# CMPUT 291 – Mini-Project 2

Created By:

Raamish Naveed
Weichen Qiu
Solomon Song

# General Overview and User Guide

Our program is a command line application that works just like a forum. It allows people to view user reports, post questions, post answers, vote on posts, search for posts, etc. A program like this has a lot of benefits. Firstly, a platform like this encourages discussion. This is the main objective. As we dive deeper, we see that collaboration and communication can be improved as well. With older platforms such as emails being slow and unproductive, forums can be one way to improve communication.

When the user starts the program, they have to input a port number under which the MongoDB server is running. The program will then create three collections named Posts, Tags and Votes based on the .JSON files provided. Terms from each post's body, title and tag are extracted for faster searching in the next part of the program. This is Phase 1, where the document store is built. In Phase 2, we will be operating on the document store. The user will be given the option to enter in their user id and view a report. However, this is not mandatory. The user can also post a question, search for questions, answer a question, view all the answers to a specific question, and vote on a post in Phase 2.

We have implemented some validation features to ensure that the input is correct. However, it is highly recommended that the user double checks their inputs and selects the correct instructions on the menus to ensure a smooth run.

# Software Design

Our program runs on python and implements the Pymongo library. Our code is structured into two main files for phase 1 and phase 2, where we files is separate into specific functions for each action. Our program is linear and procedural. Our program flows from phase1.py to phase2.py. These are the files needed to run the program:

- phase1.py
- phase2.py
- Posts.json
- Tags.json
- Votes.json

Phase1.py: This is the starting point of this application. The user is required to input a port number under which the MongoDB server is running. Next, the program will create three collections named Posts, Tags and Votes based on the .JSON files provided. Terms from each post's body, title and tag are extracted for faster searching in Phase 2. Once these collections are built, the program for this phase ends.

Phase2.py: The user is required to input a port number under which the MongoDB server is running in order to operate on the document store created in Phase 1. The user will be given the option to enter in their user id and view a report. The user can also post a question, search for questions, answer a question, view all the answers to a specific question, and vote on a post.

# Testing Strategy

We followed a sequential approach and tested our program manually step by step. Initially we created our document store. After all that was done, we tested our report function to see if it gave us the correct report for the user. Next, we tested the post function where we were constantly manually checking in Terminal to see if the data was actively being added to it or not. Next, we tested the search function to make sure that it works. We played around with our options and kept discovering bugs and fixed them as we went along. After that, we selected a post and tested our answer function, which seemed to work perfectly fine. We went back to the main menu, searched for another post, and decided to post a couple answers to our questions. More bugs were discovered and fixed. We followed the same approach for the remaining functions until all the functions seemed to be working fine. Towards the end, we tested scenarios where users may make an error and took preventative measures to prevent the code form breaking.

# Running Instructions

Python 3.5 is required for the program to work. Please run the following command in terminal:

python3 phase1.py <port_number>
python3 phase2.py <port_number>

The .json files must be in your program directory.
Make sure to change the port number in the command above.

# Group Work Break-Down Strategy

We created a repo on GitHub to ensure everything was accessible to all of us. Along with that, all of us kept separate backups of our files on our personal computers to ensure nothing was accidentally lost or deleted. We used a group chat to coordinate and make sure everything was going smoothly. We organized a group meeting on call to split up tasks and follow up meetings every few days to ensure that everyone was on track. We thought it would be fair if everyone did an equal amount of work, that's why we tried to split up tasks equally. Roughly, each of us spent around 20-25 hours in total on our separate parts of this assignment. Person responsible for the following tasks:

Weichen Qiu:

- Setting up GitHub repository
- The full Phase1.py file
- Coded the base structure for the menu and did the report function
- Additional testing
- Fixing Bugs

Raamish Naveed:

- Post function in Phase2.py
- Search function in Phase2.py
- Additional testing
- Fixing bugs
- Writing the Design Document

Solomon Song:

- Question action-Answer in Phase2.py
- Question action-List answers in Phase2.py
- Question/Answer Action-Vote in Phase2.py