Natnail Ghebresilasie, Afaq Nabi, Krutik Soni

Mini-Project 1 Report

General Overview:

When first opening the program after inputting a correct port number, a user is greeted by a login screen in which they are able to provide a userID to log in with and in doing so are provided with a summary of the number of questions they have posted, the number of answers, the number of votes they have made, and the mean score for each questions and answers. Otherwise, they can simply say no to the login and continue to the actions that are allowed. All users are allowed to either post a question, search a question, or exit the program. If they choose to post a question, users are able to provide a title, body text and some tags to attach to the post. The post is then inserted into the database called 291db where the records are saved. Users are able to answer a question, vote on an answer, or vote on a question. However, none of these tasks can be done without first searching for a question. Users, provide the program with keywords split between spaces that they would like to search with. The program then retrieves the correct posts that are questions with the matching keywords in the terms, tags, title, or body field. Users can select a question and are able to see a summary of it., In doing so, they are allowed to vote on the chosen question, pick an answer if the question has any and see the summary for those, or vote directly on the question. If picking an answer is chosen, they are able to see the summary for the answer and then choose to vote on the question. For a representation of the flow of the program, look at figure 1 on the last page of this report document.

Design:

The main function is where the login process takes place. It allows the program to continuously run until the user chooses to exit the program completely in which case the loops would terminate closing the program. Within the loops is where the functions such as User_Id are called which returns the userID entered by the user initially at the login screen. The functions questions, answers, votes are in charge of handling and finding all the questions, answers, and the votes that a user has created with the parameter userID passed into it to use to search the database. Function main_search is in charge of handling, calling, and returning results from the function search_questions. Keywords entered by the user are searched in the search_questions function in which the terms index initially created in phase 1 is used. These results are then sent into the main_search function that handles them and stores them to be later printed. Users are able to pick a question from the displayed question, if a question that is not displayed is entered the user is prompted to enter the correct question ID from the displayed results. This is done through the function pickQ. Questions will be used in this function and if a correct question is selected, the ID is then sent into the function pick display_picked_post where the summary of each question is handled and displayed. Users who select to see the answers for a question that has answers are able to see all answers printed out that are linked to the question, this is handled by the function pick_answer and display_picked_post. The function post_question is in charge of handling the duty of when a user decides to post a question from the login screen. It takes UserID as a parameter if one was provided initially when first logging in, and variables question_Id and tags_Id that are in charge of creating unique IDs for new question posts and tags to be inputted into the database. search_question is one of the largest functions and is in charge of handling the searches for the keywords the user provides that is then passed in as a list parameter to the

function. The function iterates through each keyword in the list and then returns the results of the search removing the duplicates for searches that have a term and tag that are the same such as "iphone." actionAnswer is in charge of handling a new answer that is posted to a post, it takes parameters UID which can be NONE if the user initially doesn't provide one, parameter answer_Id is the variable that handles creating a new unique ID for the new answer that will be inputted into the database. Parameter postID is the ID of the post in which the answer will be posted on. The function will query to post the answer and then will query the for the question to increment the count field. actionVote will allow the user to post on a question or answer and will increment the score field by 1. It takes a parameter UID in which it is used to check if the user has already voted on a post. Parameter postID is the ID of the post to vote on and votes_Id is the variable that handles creating a new unique ID for the vote to be inserted into the database.

Testing:

The testing strategy our group decided to use was we would create each function needed for the program and then unit test each function to make sure they behave as intended. Majority of the functions were simple queries and the biggest hurdle was understanding mongodb syntax in python. A lot of errors discovered were simple syntax errors in which the fix was very easy to make. Unit testing allowed us to make sure the functions behaved properly before inserting them to the final program which saves us from a lot of headaches and problems. Writing the functions on a smaller scale and individually testing them was the main strategy we used for testing. We used MongoDB compass to see the database in a clearer way and better monitor it.

Group Work Break-down:

The project was mainly done on Visual Studio Code that has a liveshare feature in which many people can code and work on the same files together. Afaq Nabi was in charge of creating phase 1 and the main function of phase 2, while Krutik Soni and Natnail Ghebresilasie were in charge of the functions in phase 2 and report. Majority of the time however we simply made the functions together, working on them together let us make sure we covered as many constraints as possible as 3 people were on it. On average we spent 5 hours a day on this project roughly starting around 7pm everyday until midnight. Total hours spent was roughly 40-45 hours and all group members contributed equally and completed all the necessary work they were tasked with to the creating the mini-project.

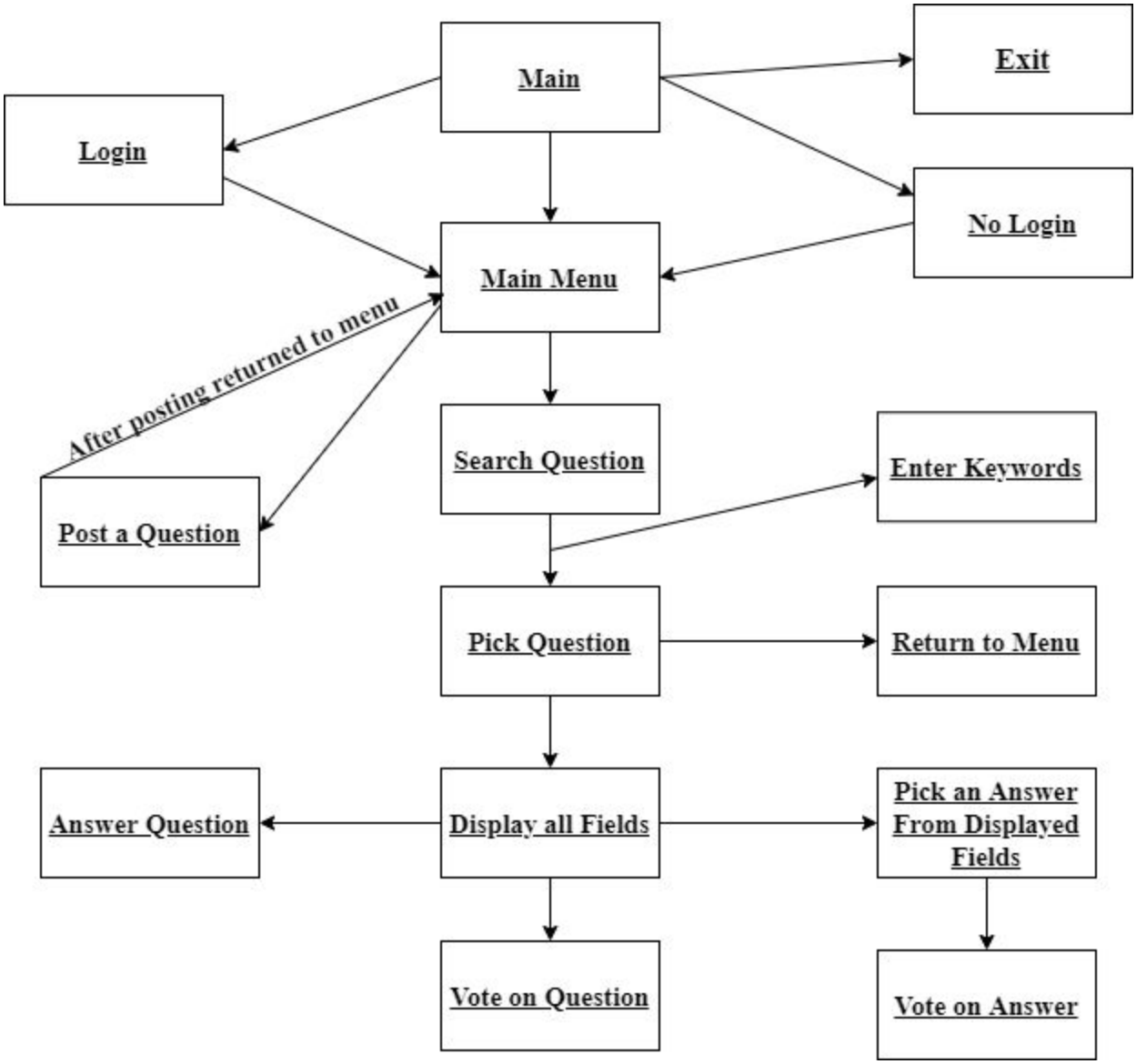

Figure 1: Flow of data between components