Sezvo-ndinemwi Mpfunya (mpfunya), Aidan Kucharski (akuchars), Nikita Adekar (adekar)

CMPUT 291 Mini-Project 2 Report

Overview

The goal of this project was to gain experience with working with data stored in files and NoSQL databases. A document store system was designed using MongoDB for operating on files and indices. Mini-Project-2 is a python command-line application that interfaces with the MongoDB Python 3 package (pymongo). Using the program users can perform searches and updates based on the current menu options.

The database schema for this project is defined as:

· Posts(Id, PostTypeId, AcceptedAnswerId, CreationDate, Score, ViewCount, Body, OwnerUserId, LastEditorUserId, LastEditDate, LastActivityDate, Title, Tags, AnswerCount, CommentCount, FavoriteCount, ContentLicense)

· Votes(Id, PostId, VoteTypeId,UserId, CreationDate)

· Tags(Id, TagName, Count, ExcerptPostId,WikiPostId)

User guide/Detailed design

Phase 1 (database creation)

Main - The main function acts as the point of database compilation for the program.The application is opened using the command "create_database.py". Once this is successful the system will immediately give the user a prompt to connect to port.

Port Number - Users are required to input a number with 5 digits. If input is incorrect an error message will appear and the prompt will reappear. Once the port connection is successful Json files with the specified collection names, Posts, Votes and Tags for the database are uploaded into a database called " 291db". After a successful creation the user is shown the time it took to create the database.

CMPUT 291 Mini-Project-2 Report

Sezvo-ndinemwi Mpfunya (mpfunya), Aidan Kucharski (akuchars), Nikita Adekar (adekar)

Phase 2

Main – The main function acts as the point of execution for the program. Database input is located here and specifies actions to perform for each input type. The application is opened using the command "mini_project2.py " Once this is successful the system will immediately give the user a prompt to connect to a port that must be of length 5 digits. If input is incorrect , the user is prompted to enter again. A successful connection will connect to a database named 291db on the server. Users are asked to enter their user Id or to skip.

skip - at log-in time, if selected users will not be given a user id and will be given a status as an anonymous user.

If a user id is provided, the user will be shown a report that includes (1) the number of questions owned and the average score for those questions, (2) the number of answers owned and the average score for those answers, and (3) the number of votes registered for the user. Users may also use the system without providing a user id, in which case no report is displayed.

Menu Option breakdown

post_a_question – The user is able to post a question by providing a title text, a body text, and zero or more tags. A unique id is assigned to the post by the system, the post type id is set to 1 (to indicate that the post is a question).The post creation date is set to the current date and the owner user id is set to the user posting it (if a user id is provided). The quantities Score, ViewCount, AnswerCount, CommentCount, and FavoriteCount are all set to zero and the content license is set to "CC BY-SA 2.5".

search_for_question - The user is able to provide one or more keywords, and the system will retrieve all posts that contain at least one keyword either in title, body, or tag fields. For each matching question the title, the creation date, the score, and the answer count are displayed. The user is then able to select a question to see all fields of the question from Posts. After a question is selected, the view count of the question increases by one (in Posts) and the user is able to perform a question action

question_action_answer - The user is able to answer the question by providing a text. An answer record is inserted into the database, with the body field set to the provided text. A unique id is assigned to the post by the system, the post type id is set to 2 (to indicate that the post is an answer), the post creation date is set to the current date and the owner user id is set to the user posting it (if a user id is provided). The parent id is set to the id of the question. The quantities Score and CommentCount are all set to zero and the content license is set to "CC BY-SA 2.5".

List_answers - The user is able to see all answers of a selected question. If an answer is marked as the accepted answer, it is shown as the first answer and is marked with a star. For each answer, the first 80 characters of the body text (or the full text if it is of length 80 or less characters), the creation date, and the score are displayed. The user is able to select an answer to see all fields of the answer from Posts. After an answer is selected, the user may perform an answer action.

add_vote - The user is able to vote on the selected question or answer if not voted already on the same post (this constraint is only applicable to users with a user id; anonymous users can vote with no constraint). The vote is recorded in Votes with a unique vote id assigned by the system, the post id set to the question/answer id, vote type id set to 2, and the creation date set to the current date. If the current user is not anonymous, the user id is set to the current user. With each vote, the score field in Posts will also increase by one.

Testing Strategy

Our testing strategy was to use data provided in the json files. Some tests were done as follows:

- Verifying that a successful connection was made.

- Verifying that the data was added into each table correctly.

- Ensuring that unique ids were added for specific commands

- Input testing was added for each command. Returning a message when the input was invalid.

The code was modified to accommodate these needs.

Group Work Strategy

Aidan Kucharski (akuchars)

• Hours: ~15 hours

• Progress: Provided most of the groundwork and organization for the project. Defined the project structure and developed functionality for creating the database in Phase 1 and building the main for Phase 2.

Nikita Adekar (adekar)

• Hours: ~15 hours

• Progress: Developed functionality for posting a question, searching for questions, and adding an answer. Also contributed by fixing bugs and writing unit tests for the system.·

Sezvo-ndinemwi M'pfunya (mpfunya)

• Hours: ~14 hours

• Progress: ~ Developed functionality for listing answers and adding a vote. Also contributed by writing the project report.

Work Coordination

We coordinated through GitHub. An organization was set up for the project (Group-CMPUT-291-Databases) and each member added their contributions with a new file to track progress. Members notified the group when updates were being made and this created a good workflow with minimal issues. Contributions from each member were compiled into one py file.

Communication was performed via Slack and members used this platform to ask questions and to clarify additions to the work. All the work was done remotely since members were in different locations.

Design decisions

All members were familiar with python and found it to be the most suitable for executing the project. Using python allowed the use of libraries such as datetime which were convenient for setting the times for posts. Pytest allowed us to continually debug the work as it was being built which made identifying key flaws easier.

The whole group agreed that if a user was anonymous user id would be ‘’ otherwise no other major design decisions were made.