COP290 Lab-3

Fouriers

1 Overview

queriKorner is a question-and-answer website that allows users to ask and answer questions. Users can search relevant questions and can filter all questions by tags. Questions and answers can be sorted based on votes and time. Users can also upvote and downvote questions and answers. Users can see their profiles and can edit their details. Users can also see top users and top questions. Each user has a reputation score calculated based on the votes on their questions and answers.

2 Baadal VM link

3 Site Map

4 ER Diagram

5 Github Link

6 API descriptions

The following are the APIs that we have implemented in our project:

- /
  This renders the home page of the website. It contains the list of recently asked questions which is fetched from question database. It also contains a truncated list of top users and top questions which is fetched from users and questions database.
- Authentication APIs
  - **/login**
    This is the login API. It takes values of (Username , Password) from the user and checks if the username and password are valid or not. If the username and password are valid, then the user is logged in and redirected to the home page. If the username and password are invalid, then the user is redirected to the login page again.
  - **/signup**
    This is the signup API. It takes values of (Username , email , password) from the user and creates a new user in the database. It also checks if the username is unique or not and if the email is valid or not.
  - **/logout**
    This is the logout API. It logs out the user and redirects to the login page.
- Tag related APIs:
  - **/tags/<string:sort>**
    This API fetches all the tags from the database and displays them in the form of cards and applying Pagination. The tags can be sorted based on the number of questions (QCount) and lexicographically (TagName).
- User details related APIs:

- **/users/<string:sort>**
  This API fetches all the users from the database and displays them in the form of cards and applying Pagination. The users can be sorted based on the reputation score (RepScore) and lexicographically (UserName).
- **/myprofile**
  This API fetches the details of the logged in user from the database and displays them in the html page.
- **/editprofie**
  This API fetches the details (UserImage , UserLocation , UserEmail) of the logged in user from the form submitted by the user and the values are updated in the database.
- **/changepasswd**
  This API fetches the details (OldPassword , NewPassword , ConfirmPassword) of the logged in user from the form submitted by the user and the values are updated in the database if the OldPassword matches with the password in the database and the NewPassword and ConfirmPassword match.
- **/otherprofile/<int:id>**
  This API fetches the details of the user with the given id from the database and displays them in a html page which shows limited information.

- Question related APIs
  - **/questions/<string:sort>**
    This API fetches all the questions from the database and displays them in the form of cards with Pagination. The questions can be sorted based on the number of votes (QuesVotes) and lexicographically (QuesTitle).
  - **/question/<int:id>**
    This API fetches the details of the question with the given id from the database and displays them in a html page.
  - **/ask**
    This API fetches the details (QuesTitle , QuesBody , QuesTags) of the question from the form submitted by the user and the values are inserted in the database.
  - **/answer/<int:id>**
    This API fetches the values of the answer from the form submitted by the user and the values are inserted in the database. The id refers to Question Id to which the answer is being posted.
  - **/search**
    This API fetches the search query from the form submitted by the user and relevant values are searched in the database and displayed in the html page.

- Voting APIs
  - **/upvoteAns/<int:id>**
    This API fetches the id of the answer from the url and increments the value of AnsVotes in the database of answers , answervotes and user reputation .
  - **/downvoteAns/<int:id>**
    This API fetches the id of the answer from the url and decrements the value of AnsVotes in the database of answers , answervotes and user reputation .
  - **/QuesUpvote/<int:id>**
    This API fetches the id of the question from the url and increments the value of QuesVotes in the database of questions , questionvotes and user reputation.
  - **/QuesDownvote/<int:id>**
    This API fetches the id of the question from the url and decrements the value of QuesVotes in the database of questions , questionvotes and user reputation.

7 CI/CD Test Coverage

8 Design decisions

- We decided to use Bootstrap 5 instead of other tools such as ReactJS to make front end of the website because importing and using ReactJS would have made the website heavy and slow and importing Bootstrap 5 with CDN is much easier and faster.
- We used `spacy` for natural language processing API for search api instead of other APIs such as NLTK because we found spacy easier to use.
- We decided to use datetime module of python to print time in format such as "2 minutes ago" instead of directly using string . We also used python re module to check if the string is a valid email or not instead of relying on html for it.
- We used module `flask_paginate` to use pagination in all users and all question pages `flask_bcrypt` to hash passwords and `flask_login` to manage user sessions.
- We assigned a limited number of characters in the title and body of the question and answer to prevent the user from entering a very long string which would have made the website look bad. This made some data from kaggle dataset unusable.
- Instead of storing whole images in database we stored a image link in database .
- We decided to use seperate tables for (questionUpvotes) and (answerUpvotes) instead of storing them in the same table because it was convinient

9 ML API

We have used Spacy which is an advanced natural language processing library to sort results on the basis of relevancy. The method takes user's query as an input and tokenize it. Similarity Score of query is calculated with all the questions in the database which is a value between 0 and 1. The score is calculated using the cosine similarity metric, which is a measure of the similarity between two non-zero vectors in a high-dimensional space. The API then sorts the relevant questions in descending order of similarity score.

9.1 Cosine Similarity

Cosine similarity is a measure of similarity between two non-zero vectors of an inner product space that measures the cosine of the angle between them. The cosine of 0° is 1, and it is less than 1 for any other angle. It is thus a judgment of orientation and not magnitude: two vectors with the same orientation have a cosine similarity of 1, two vectors at 90° have a similarity of 0, and two vectors diametrically opposed have a similarity of -1, independent of their magnitude. Cosine similarity is particularly used in positive space, where the outcome is neatly bounded in [0,1].

9.2 Using SpaCy for natural language processing

Spacy is an open-source software library for advanced natural language processing, written in the programming languages Python and Cython. We used `nlp = spacy.load("en_core_web_md")` to load a medium english core. And using data from (QuesTitle) in questions database we trained it to give similarity score with user's query.

10 Unique Features

10.1 MathJax & Quill Text Editor

We have used MathJax to render mathematical equations in the questions and answers. We have also used Quill text editor to make the text editor more user friendly.

10.2 Password Encryption

We have used `flask_bcrypt` to hash passwords and `flask_login` to manage user sessions.

11 Challenges

12 Limitations / FutureWork

2 fa email verification forgot pass search + tagsearch offensive text recognition moderator access – delete users edit/delete questions/answers comments

13 Video Link

bvdhvbgafkavn

14 Token Distribution