Project Report: MINI-Stackoverflow

Anupam Kumar(18110022), Vivek Modi(18110190)

Introduction

Stackoverflow is a public platform where one can ask questions and answer them. Our goal is to make a mini version of the stackoverflow with its core features. Our project has three components. The first is database design, second is the server which listens to client requests and processes database related read/write functions. The last one is the client itself.
We started the project by keeping in mind that providing the maximum number of features using our database. Therefore, we need to select a user-friendly and easy to use database i.e., MySql. We initially implemented a pickle database and cli based server but it did not perform well with many features. Therefore, we tried to make a new MySql database and react based front end client and flask based python server.

Work Methodology

Mini Stackoverflow is a portal where any user can ask questions and any other user can answer which is publicly shown to everyone. A new user creates a new account with User Id, name, email-id and password. After sign up, the user will be redirected to the user login page where they can enter their login credentials. After successfully verifying these credentials, the server provides a unique token to the client. This token is stored in localStorage by the user. For all subsequent write requests, this token is used as an authentication token. Once a user logs out, the token is deleted from the browser. On every login, a new token is generated from the server. This token is stored in the database along with the userid. This enhances the authentication part of the web application. The client will then be redirected to the home page where they can see the questions. The questions will be displayed on the basis of the number of upvotes.
In the question page, we can answer the question, upvote the question, edit or delete the question(only by the author). In the answer section, the client can upvote the answer as well as edit or delete the answer. The client can post a new question with three required parameters - Question title, question description, and tag. We can also search a particular question based on the tag given by the author.

We have used a React based application as our frontend client. It uses the axios library to execute HTTP/1.1 GET and POST requests. It serves as a multipage website providing different pages such as Home page, Question page, Add new question page, Profile page, Signup/Login Page, etc.

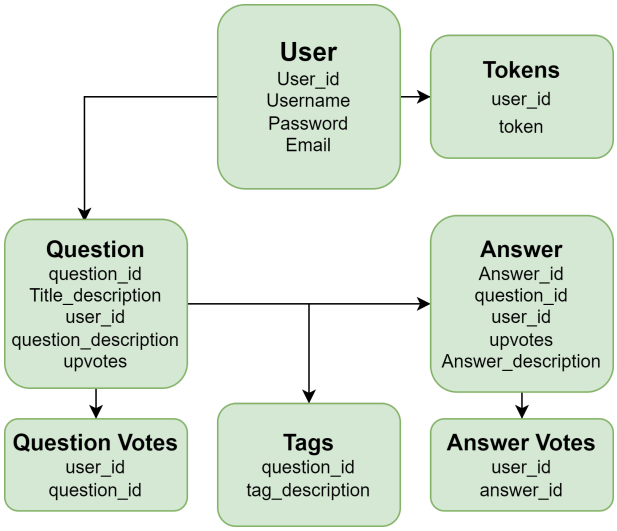
Tools used

- Python: for flask server
- MySql: for database
- Reactjs: for client and front end development
- React-bootstrap: front-end framework for design
- Axios: for making HTTP/1.1 GET and POST requests

Features

- Signup
- Login
- Logout
- Post Question
- Answer Question in a thread
- Add Tag
- Delete answer
- Delete question
- Delete tag
- Search question using tag
- Upvote question/answer
- Reputation
- Profile page

Database Design



We have used a MySql database. You can see the database schema. There are seven tables interlinked with each other by certain foreign keys. User table has User_id, username, Email, and password in an encrypted format using SHA256 hash algorithm. User_id id the foreign key for Tokens table, Answer Votes table, Question Votes table, Questions table, and Answers table. Question_id is the forign key for Answers table, Question votes table, and Tags table. Answer_id is the foreign key for the Answers Votes table.

Evaluation, Testbed configurations, data trace/test vectors considered

Avg. login time(in ms)	Avg. add question time(in ms)	Avg. delete question time(in ms)	Avg. add answer time(in ms)	Avg. add answer time(in ms)
20.441	11.002	11.3588	10.986	13.5802

Summary

We have successfully built a Mini stackoverflow with certain features mentioned above. We learned pros and cons of using different databases. Considering the type of queries by user, using MySql seemed to be the best choice for us. We learned about deploying a flask based server. It helped us in writing get and post methods of the server in a structured way. We also learned how to build a React based frontend application, where we learned how to process GET and POST requests.

Future Extension

- Ability to flag questions under certain categories based on spam, duplicate, rude/abusive comments, etc.
- Adding question moderation(showing approved questions)
- Ability to search using question title
- Adding timestamp in question as well as answers
- Implementing HTTPS or HTTP/1.2 to provide better security in server-client communication
- Implementing question moderation and rest of the features in the frontend.
- Making the server multithreaded

Links

- Presentation Slides: https://docs.google.com/presentation/d/1VZZLL69tUmLIR-QFgzcZ6NibzoZS2WVodG24YDB3RcY/edit?usp=sharing
- Github repository: akcgjc007/CS433-Project