DBMS-Application

How the application was built

The application development process involved downloading the Software Engineering dataset from the Stack Exchange Archive. The downloaded XML files were parsed using a Python script. Then, `init.sql` was created using a pre-existing DDL, and various integrity constraints were added. The data was inserted using `data.sql`.

After the database was created, the backend was developed by creating various controllers, models and routes. The frontend was created in parallel using NextJS. Authentication was added along with the middleware after this, and the frontend was finally integrated with the backend.

Overall system architecture

Backend

- The `backend` folder consists of `controllers`, `db`, `middleware`, and `routes`, in addition to files like `./index.js` and `models.js`.
  - `controllers` has several files for the various backend APIs/functions. Examples of these include the APIs for creating answers (in `answer.js`), for signing in (in `auth.js`) and for autocomplete (in `autocomplete.js`).
  - `db` has various database related files. `init.sql` defines the DDL, `data.sql` inserts data into the created tables, and `drop.sql` drops the created tables. The problem-statement also asked for accounts to be created for pre-existing users, with a username and the default password as the username. For this, `gen_cred.py` populates the *credentials* table with usernames (display_name concatenated with user ID) and passwords (same as username).
  - `middleware` contains a single file with the API verifyToken. This API checks if the jwt is valid, and returns responses checked by the frontend to trigger redirection to relevant URLs
  - `routes` contains the various backend routes. The various functions are imported from the controllers, and used in either put, delete, post or get methods.
  - `index.js` brings everything together, and runs the backend.

Authentication Authentication is mainly done in the controllers (for the SignIn and SignUp APIs) and middleware in the backend. When a user signs up, a hashed password is stored, and the user is provided with an accessToken. This token provides the session. Logging out deletes the session. The user can log in with the correct credentials to get an accessToken.

Frontend

- The `frontend` consists of `pages` and `components`

Pages

- create/answer/[qid].js: This page allows the user to add an answer to an already existing question. The question and all the previous answers are also displayed.
- create/question/[userid].js: This page allows the user to create a question with tags, description and other details.
- edit/answer/[answerid].js: This allows the user to edit an already existing answer. The previous answers are already displayed on the editor.
- edit/question/[qid].js: This allows the owner of the question to edit details and tags. One must note that the previous tags of the questions are shown. But the one must include the previous tags while editing to endure they are not missing.
- posts/[postid].js: This page shows the question, along with its details and the subsequent answers. It allows the creator of the question to edit the question, and the people who answered the question to edit their answers.
- tags/[tagname].js: This page shows the top five tags.
- dashboard.js: This page shows the details of the user who has logged in and the questions he/she has created. The page also gives options to the user to edit or delete questions.
- explore.js: This page allows users (even guest users) to serach for posts, linked to multiple tags (or) created by a particular user. This options can also increase the search result-set, and sort the results by score and time.
- signin.js: Allows the user to login. Note that a cookie containing the access token is created. The access token allows the user to be logged in for a maximum of 24 hours.
- signup.js: Allows guest users to sign up and create an account. Creating an account enables the individual to create, edit and delete questions. It also allows the individuals to give and edit answers to previously existing questions. The password for the new user is the username by default.

Components

- autoTags.js: A form component using and that fetches values from the database based on present tag values. The presently selected tags are shown in the as .
- autoUsers.js: A similar component to autoTags, but modified for users.
- editor.js: A text-field like component that allows people to easily post answers and questions with necessary formatting.
- sidebar.js: Provides a side navigation panel that allows individuals to access the best questions and tags. Also allows a quick way to create question to authenticated users.
- userDetails.js: A -like component providing details of a user. Is used in question-posts, answers etc.

Programming Languages

- **Frontend** - The frontend of this project has been written in Next.js, which is an open source web development framework which extends the functionalities of React based webpages. This project has also been supported by MaterialUI which helps us import ready-made components.
- **Backend** - The backend of this project was written using ExpressJS, which is a backend web application framework for building REST APIs. SQL was used for the database, and Sequelize was used to integrate Javascript with the database. Several other libraries were used, like `cookie-parser` and `bcrypt` for authentication.

Group Member Contributions

- Kushagra Gupta (MistyRavager): (*frontend*) Authentication, Dashboard and Create/Delete Questions
- Rahul Ramachandran - (*backend*) Various backend APIs, Authentication
- Rishit D (purplehand52): (*frontend*) Autocomplete, Search/Explore, Edit Question
- Suryaansh Jain - (*backend*) Python scripts, Various backend APIs, Models