Devam Punitbhai Patel

Dns682

11316715

Project

Design-Report

Installation steps:

Please start a react app using command

npx create-react-app anyname –use-npm

start anyname

copy the code in my src directory to anyname's src directory

in the root directory run commands

npm install mysql, npm install express and npm install cors.

Then do docker-compose up to start the backend server.

Software-Architecture:

For my Project I have used client-server architecture which includes React application for the frontend (Client-side) and nodejs with express framework for the backend (server-side) and MySQL for the database.

Using MySQL seemed to be a reasonable choice for the database to me because I wanted a relational database management system because most of my elements or entities were related to each other.

In my system I used 1 database and 6 tables: users, channels, posts, comments, likes, dislikes.

users table:

I used it to store information related to a user using the system. I stored things like username (unique identifier), firstName, lastName and password.

channels table:

All the channel related information is stored in the channels table. I also have a column for createdBy which stores the username of the user that created the channel.

posts table:

All the post related information is stored in the posts table basically I have extended the simple message application to a post application by just adding a topic to the message that the message was about. I

also have a column for createdBy which stores the username of the user that created the post and channel_id to store the id of the channel that the post was created on.

comments table:

All the comments related information is stored in the comments table. I also have a column for createdBy which stores the username of the user that created the comment and c_id to store the id of the channel and post_id to store the id of the post that the comment was created on. I also have a column for parent_id to store the id of the parent comment for nested comments.

likes and dislikes tables:

likes and dislikes tables are pretty much the same they are used to store likes and dislikes on a post or somment. They store information like post_id, comment_id and the username. I had to create 2 tables for them because a user can even dislike a post without ever liking it so I can't just remove the like from the likes table.

Npm packages used:

I have used npm packages:

Express: for APIs.

Mysql: for using the database

Cors: for frontend and backend connection.

For the frontend I have used react-router-dom for dom tree manipulation.

I also tried using bcrypt for password hashing but it was giving a errors so I decided to skip it for now.

Basic-design idea:

The application starts at a login page can also be called a landings page and you are given options to signup or login. On clicking signup, you are taken to the signup page (done with <Link /> component).

And authentication is used so login is required to use the system because authentication is set in the Login component, and it is checked when using Link to routes.

Then you are taken to the channels page where you can see the channels already in the database and get an option to create one yourself. You also get the show profile option which takes you to your profile page and your profile information is displayed with a button to change your password. If you are in the admin account, you also get an option to delete channels and an extra option that takes you to the admin dashboard where you can see all of the users currently on the system along with their contributions. Contribution is one of my systems extra features (when you create a post you get 10 contribution points and when you post a comment you get 5 contribution points. You also get assigned a badge based on your contributions which you can see on show profile.

The channels page also has a search component which gives you 2 options to choose from: whether you want to search for content on the system or user related information. For user related information you get 5 options to choose from: mostPopularUser, minPopularUser, mostCBUser (User with maximum contribution to the system), minCBUser and User: (username, to search for content by any specific user with its username).

The Search component displays a page with list of posts related to the string you entered or user information if you searched about keywords for user and a list of posts created by a user if you searched by its username.

Channels are displayed in form of cards and by clicking on a card all the posts on that channel are displayed and you get an option to add more posts to it.

Then posts are also displayed in form of cards on a channel and by clicking on a read more on a post you are taken to that post where you can see comments, likes on that post you can also add likes or dislikes.