Design Report: Database and React Application

DATABASE:

- Mysql : im using mysql because of the popularity and its reliablity created tables to store data relating to them(i.e users,channels,messages,upvotes, downvotes)

SERVER: i have various packages im using see below for justification

- Express : im using it to create my server and define routes and handle requests from my app
- CORS:It assists in addressing security issues with requests from different origins.
- COOKIE-PARSER:Cookies that are associated to the client's request are parsed by cookie-parser and made available in the req.cookies object. It makes using cookies in Express easier.
- Express-session : i enforced the fact that only users who are logged-in can use the app, i used it because express-session offers session management. It enables you to keep user state between requests and retain session data on the server. User preferences and authentication information can be stored in sessions.
- Body-parser:The body of incoming requests is parsed using body-parser. It opens up the data in req.body after extracting it from the request body. I needed this alot when handling all the different type of requests.
- Bcrypt: as part of best practice and for security reasons i decided to hashpasswords before i store them in my database
- Multer : part of the requirements was to be able to have profile pictures and also post screenshots so i needed a way to store images in the website so i used multer because multer is a powerful tool that facilitates the management of file uploads in online forms. Its primary function revolves around configuring the upload object to effectively store uploaded files in a designated destination folder, such as "uploads/" for instance.

REACT-APP:

- Yes it was required to use reactjs , but i also prefer it as its architecture based on components, which provides modularization through reusable components and organisational advantages that encourage maintainability. Its virtual DOM is effective in reducing browser updates, which enhances performance. The vast ecosystem, which includes libraries and tools, accelerates development and makes difficult tasks easier.. Together, these characteristics improve overall comprehension and development efficiency by creating a codebase that is scalable, well-maintained, and effectively developed.
- A few of the tools i used include hooks like Usestate and Useeffect for handling data ,Usenavigate for switching between pages , uselocation for sending data as states between pages etc
- Also i made use of axios that lets me make various http requests like Get, Post, etc

- Also i made use of session storage to store insensitive user information across the page

Docker : Docker Compose has been employed to facilitate the process of containerization, guaranteeing a uniform setting throughout various stages such as development, testing, and production.

In conclusion

The project uses effective packages like CORS, cookie-parser, express-session, body-parser, bcrypt, and multer, and it uses MySQL for dependable data storage and Express for server-side logic. Front-end ReactJS uses essential tools like useState, useEffect, axios, and session storage to guarantee efficiency and modularity. For a consistent development environment, containerization is streamlined with Docker.my project ensures a balance between security, scalability, and development efficiency.