REPORT

A short 2-3 para summary of how you built the application :

We built our application after careful observation of the database. After a few discussions, we decided to make our application using **React(Next.js)** with a database server as Node.js. After assigning the roles, we learned **Html, CSS, Javascript, and a bit of React**. We decided to use **Figma for designing** our frontend pages.

After analyzing the requirements, we made three pages: a **Home-page**, a **login page**, and a **sign-up page** .

One of our group members with web background was interested in using **Chakra-UI** for the front end of our application. We decided on **Next.js** as it enables us to create full-stack web applications with **pool package** for connecting to our backend with the database(Postgres). We wrote our **backend logic in Node.js**. We also used **JSON web token to encrypt** our id, which we store in cookies for authentication. Our sensitive data comes from the **local.env** file.

Overall system architecture :

WEBPAGES :

Home page : The front page of the application used for user interaction.

Login page : The page for authenticating the user.

Sign-up page : The page for registering new users.

display_query page : The page for displaying the query results.

profile page : The page shows all the user information for a user_id.

post page : displays all posts for a given post_id .

Middleware : verifies cookies(JSON web token) with a local environment secret key .

APIs :

autocompletion.js : autocompletion of tag-name and display_name .

crComm.js : create comment.

createanswer.js : create answers(post) for a given post_id .

createnewpost.js: create a new post.

createuser.js : register new user .

editpost.js: edit the post for a given post_id.

getComment.js: get all the comments for a given post_id.

getuser.js: get the user information for a given user_id.

giveanswers.js : give the answers for a given post_id(parent_id).

hello.js : displays the top 5 queries in display_query page(at the start) .

login.js: login of the user and set up the cookie.

search by tags: gets all the posts with the given tags.

userpost.js : gets all the posts for a given user_id(owner_user_id).

deletepost.js : delete a post for given post_id and user_id .

Programming languages used for different components :

DESIGNING : Figma

FRONTEND: HTML, CSS , Next.js,Chakra UI

BACKEND: Javascript, Node.js , Next.js

DATABASE: Postgres

NOTE:

Chakra UI: Chakra UI is a simple, modular, and accessible component library that gives the building blocks that is needed to build React applications. Chakra UI

provides more components, improved styling API, accessibility, and intuitive component APIs than Theme UI.

Next.js: Next. js is a React framework that enables several extra features, including server-side rendering and generating static websites.

Figma: Figma is a powerful web-based design tool that helps you create anything, websites, applications, logos, and much more

---

Contribution of each group member in the implementation :

Team members:

Suraj Kumar AI21BTECH11029,

Shivanshu AI21BTECH11027,

Karthik AI21BTECH11024 .

Suraj Kumar:

1)integrating APIs with the frontend and checking the behavior of rendering components(login page, signup page, etc.) using Next.js and Chakra-UI.

2)creating cookies at the time of signup(Middleware).

3)edit and delete post content/tags(API).

4)CSS of search query component and post pages.

5)Fetching upvotes, downvotes, and views for every post.

6)Javascript of all the components.

7)managing the GitHub repository and git.

Shivanshu:

1)Designing of pages using Figma.

2)CSS of Home page, Display_query page, profile page, and post page.

3)Registering a new user(API).

4)Fetching the comments for every post and creating a new comment for a post.

5)CSS of Qna, Search Here, Nav-Bar, Right-side Bar, and Login components.

Karthik:

1)Analysing the Database.

2)CSS of Sign-up and Login pages.

3)autocompletion for tags and display_name(APIs).

4)Search Posts(using user_id ,tags.)(APIs) and Create Posts.

5)Fetching all the answers for a given post_id(APIs).

6)Report.
