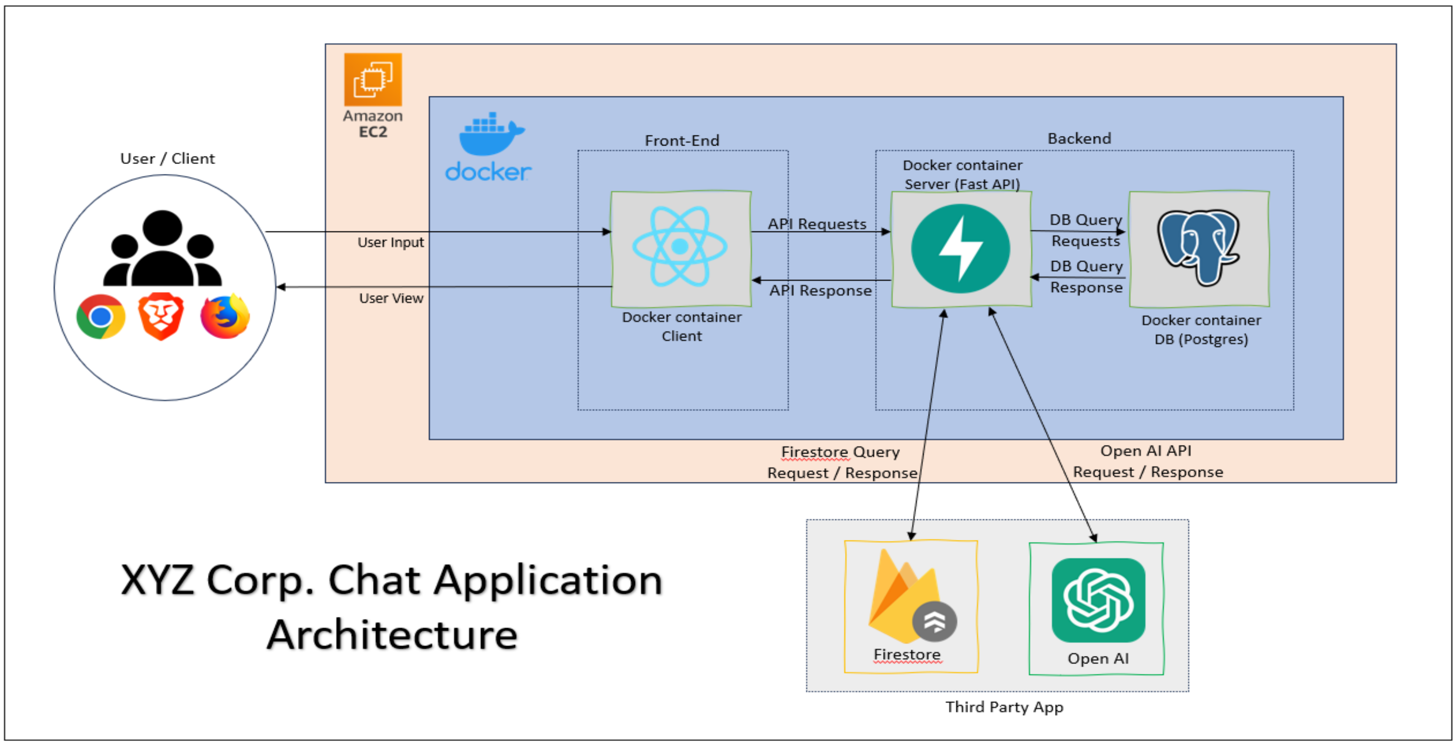

Fig 1.1 Chat application architecture

This architecture is design for a chat application. It has multiple components that interact with each other to enable the users to communicate with AI assistant of XYZ Corp. Let's Breakdown the components:

Users:

- User/Client: It is a desktop application/ mobile application like chrome, Firefox, brave on which user can interact with the application.

Front-End:

- Client (Docker container): It is where the client side code (React) of the chat application runs.

Back-End:

- Server (Fast API): It is the main server-side component of the chat application. It is responsible for business logic, controller, handling requests, routing, storing and fetching user credentials to Postgres and their messages to cloud storage firestore.
- DB (Postgres): This is the database that stores the users credentials like user name, password, user email, employee id.

Third party app:

- Firestore: This is cloud based NoSQL database that is used for storing the user chat messages (i.e. user inputs and AI assistant responses).
- OpenAI API: This is an API that allows us to access LLMs (gpt-3.5-turbo), which gives responses to the query user requested.

Communication Flow:

- User input: The user provides input or other data into the chat application.
- API Requests: The client-side code sends API requests to the server.
- DB Query / Firestore / OpenAI for chat: The server-side code queries the database to retrieve or store data first, then it will stores the user chat to Firestore and send request to OpenAI with user input, after OpenAI response it will store the response to same message collection and message collection will be returned.
- API Response: The server-side code sends an API response back to the client.
- User View: The client-side code updates the user interface to display the new data.