CONCEPTUAL-ARCHITECTURE DOCUMENT

TEAM CUPRIC SECKEL

ARCHITECTURALLY SIGNIFICANT REQUIREMENTS

1. Chat application should be distributed and peer to peer application.
2. Server should provide the centralized control for the application.
3. Application should be browser based.
4. Clients should be able to send and receive texts.
5. Clients should be able to chat with all the online peers.
6. Server should notify all the online clients, whenever any client comes online and logs out.

EXPECTED SYSTEM QUALITIES

1. Chat Application should provide secure chatting environment.
2. Application should provide ease of use to the clients.
3. Application should provide privacy for the chat clients.

Conceptual Architecture of the chat application consists of three components – Client, Server and Database.

Figure 1

~~The following picture~~ shows the conceptual architecture of the chat application with the components and their interactions.

CONCEPTUAL ARCHITECTURE

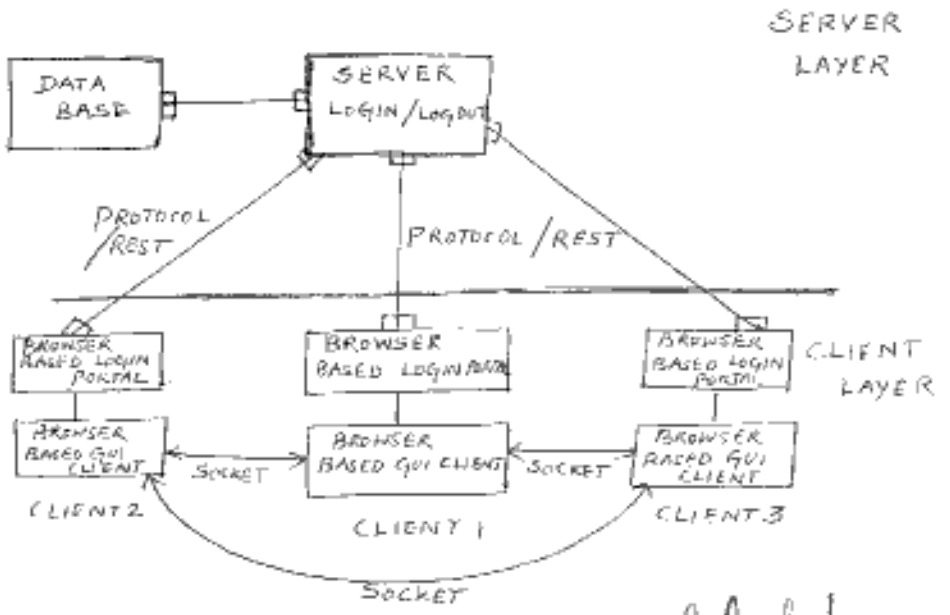

Figure 1. Conceptual Arch. for chat system.

CLIENT COMPONENT

Client Component includes Browser based login portal and Browser based GUI chat application.

Number diagram?

1. Client types in the server URL in the browser and requests for the login page.
2. Client types in the username and password into the required fields and clicks log in.
3. If new client, they have to registers with the server to participate in the chat and logs in with the credentials.
4. When the login credentials are verified, the client enters into the chat application.
5. Application GUI shows chat history, List of users online, Text area to type messages and a submit button, Logout button.
6. Once the client wants to communicate with a peer, it establishes a socket through which further communication takes place.

Is the client human or software?

7. When the client wants to go out of chat, clicks the logout button. The server disables the connection and the socket connection between the peers is also disabled.

Scenario Diagram(s)?

Responsibilities:

- Allows client to initiate chat by logging in to the application through browser.
- Allows client to send and receive texts from online peers.
- Keeps track of the chat history.

Collaborators:

- Server verifies the client with the credentials stored in the database.
- Server notifies all online clients, whenever a client comes online and goes offline.

Rationale:

- Provides ~~accept~~ independence between chat clients and facilitates distributed system.

SERVER COMPONENT

Responsibilities:

- Provides authentication of the clients.
- Allows centralized control of the chat application.
- Notifies clients about all online users and clients going offline.

single point of failure - possible. Could you design a distributed system?

Collaborators:

- Interacts with the database, to store the user information.
- Verifies the client login username and password with the information in the database.
- Client submits login and logout requests to the server.

Rationale:

- Provides centralized access to the application and control of the application

DATABASE COMPONENT

Responsibilities:

- Provides data storage - user information

Collaborators:

- Server access the database through JDBC.

Rationale:

- Provides centralized data control and storage

add these changes to Meta Arch

METAARCHITECTURE UPDATES

CONNECTORS

LINKAGE CONNECTOR-TYPE

The transport of data and passing of control between the server component and the data source component is achieved through an intermediary known as the data access connector. The thread of control passes from the server to the connector, as the program counter is incremented, and the connector is invoked by passing in the appropriate connector library inside the server component of the program. In this way the data-access connector provides linkage to the database component which the server can inspect and call upon to effect modifications to the underlying messages data that is presented.

DATA ACCESS CONNECTOR-TYPE

The data access connector is a principle between the server module and a database that is configured to hold messages from the messaging peers by submitting data transported by HTTP. The peer-to-peer approach allows multiple users to interact with each other simultaneously, however there is a centralized source that must determine priority for reads and writes to the database so synchronization of messages displayed is maintained. The type of connector used should provide an API so its usage is broadcast consistently to all components throughout the system, whether a peer or another facilitator within the overall system. The data access connector is the more appropriate type of connector for the design considerations relevant in our peer-to-peer chat application.

What if you used a DAC connector between clients? Then the server gets out of the way.