**Facebook Graph API:**

* Type System: GraphQL has a flexible and self-describing type system, allowing clients to request only the data they need.
* Single Endpoint: Clients interact with a single endpoint and request exactly the data they want, reducing multiple round trips.
* Strongly Typed: Queries and responses are strongly typed, ensuring consistency and predictability.
* Customization: Clients can define their data requirements, avoiding over-fetching or under-fetching data.
* Aggregation: GraphQL can aggregate data from multiple sources and return them in a single response.

**GraphQL API:**

* Type System: GraphQL has a flexible and self-describing type system, allowing clients to request only the data they need.
* Single Endpoint: Clients interact with a single endpoint and request exactly the data they want, reducing multiple round trips.
* Strongly Typed: Queries and responses are strongly typed, ensuring consistency and predictability.
* Customization: Clients can define their data requirements, avoiding over-fetching or under-fetching data.
* Aggregation: GraphQL can aggregate data from multiple sources and return them in a single response.

**Suggestions for Web Application:**

a.) **Login using Facebook:**

* For this feature, you should use the Facebook Graph API's authentication endpoints to allow users to log in using their Facebook credentials. This will provide you with an access token that you can use to authenticate the user for subsequent interactions with the Facebook Graph API.

b.) **Finding the number of mutual connections on Facebook between two users of the application who are logged in using Facebook:**

* After successful login with Facebook, you can use the access tokens obtained to make requests to the Facebook Graph API.
* Fetch the friends' list of both users and then compare the lists to find mutual connections.
* The specific endpoint to retrieve a user's friends in the Facebook Graph API is: **/{user-id}/friends**.

c.) **Sending and receiving chat messages to a user on the application:**

* For this feature, you can utilize GraphQL API as it provides more flexibility in defining data requirements and handling real-time data updates.
* You can design a GraphQL schema with types for users and messages and define the necessary mutations and subscriptions for sending and receiving messages.
* When a user sends a chat message, use a GraphQL mutation to store the message in the database.
* Use GraphQL subscriptions to notify the receiving user in real-time when a new message is sent to them.
* Ensure that the messages are associated with the appropriate users for secure and private communication.

In summary, you can use both the Facebook Graph API and a custom GraphQL API in your web application. Utilize the Facebook Graph API for authenticating users and retrieving their friends' lists. For real-time messaging between users, implement a custom GraphQL API to handle message interactions and notifications efficiently.