**Flow**

This flow illustrates how all components work together to provide a seamless, real-time chatbot experience with secure and state-managed interactions. The integration of Redux and Faye.js ensures that both state management and real-time communication are handled efficiently.

**1. User Access**

* **Widget (UI)**:
  + User opens the chatbot widget on a website or app.
  + The widget checks if the user is authenticated using a session token stored in **Redux**.
  + If not authenticated, the widget prompts the user to log in, obtaining an authentication token.

**2. Initial State Management**

* **Redux**:
  + Manages the user session, including authentication state.
  + Stores initial UI state (e.g., conversation history, input fields).

**3. Authentication Verification**

* **Widget**:
  + Sends the authentication token to **Botkit** via **Faye.js** over **WebSocket**.
* **Botkit**:
  + Verifies the token with its internal authentication logic or an external authentication API.
  + If authentication fails, Botkit sends an error message back to the widget; otherwise, it proceeds.

**4. User Message Input**

* **Widget**:
  + The user types a message in the widget.
  + The message, along with the authentication token, is sent to Botkit using **Faye.js** for real-time communication.
* **Redux**:
  + Updates the state to reflect that a message has been sent (e.g., stores the message in conversation history).

**5. Message Processing by Botkit**

* **Botkit**:
  + Receives the message and token via Faye.js.
  + Authenticates the user (if not already authenticated).
  + Processes the message to determine the appropriate response.
* **MongoDB**:
  + Botkit queries MongoDB for user-specific data (e.g., preferences, past interactions) as needed.
  + Retrieves or stores conversation data.

**6. External API Interaction**

* **Botkit**:
  + If the message requires external data (e.g., weather info, user account details), Botkit makes an API request.
  + Botkit includes necessary authentication tokens or API keys in the request.
* **API**:
  + Responds with the required data (e.g., weather forecast).
* **Botkit**:
  + Processes the API response and prepares a message to send back to the user.

**7. Real-time Communication and Response**

* **Botkit**:
  + Sends the response message back to the widget via **Faye.js** over WebSocket.
* **Widget**:
  + Receives the message in real-time.
* **Redux**:
  + Updates the application state with the new message and any other relevant data (e.g., updated user info, new conversation state).

**8. UI Update and State Synchronization**

* **Widget**:
  + Displays the bot's response to the user.
* **Redux**:
  + Ensures that the state is synchronized across all components (e.g., conversation history is up-to-date, the UI reflects the latest interaction).

**9. Data Storage and Session Management**

* **MongoDB**:
  + Stores the conversation history and any other relevant data after the interaction.
* **Redux**:
  + Manages session persistence locally in the widget (e.g., keeping the user logged in, maintaining conversation state).

**10. Continuous Interaction**

* The process repeats for each user message, with Redux managing state, Faye.js ensuring real-time communication, and Botkit handling logic and data processing.