A diagram of a social media system

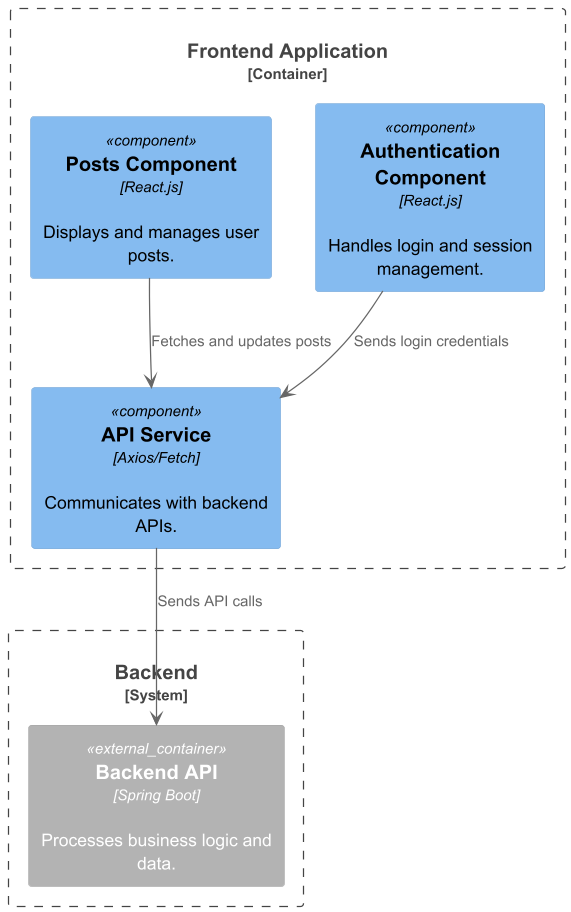
Description automatically generated

A diagram of a company

Description automatically generated

A diagram of a server

Description automatically generated



A blue rectangular object with black text

Description automatically generated

**User and Moderator Roles**

1. **User**:
   * Represents the primary actor who interacts with the social media system.
   * Activities include posting content, liking posts, and commenting.
2. **Moderator**:
   * Represents the actor responsible for maintaining the platform's integrity.
   * Tasks include managing bans, reviewing reports, and moderating content.
3. **Social Media System**:
   * The core system that connects users and moderators.
   * Enables user-generated content, likes, comments, and content moderation.

**System Context**

1. **Frontend [React.js]**:
   * Provides the user interface for interacting with the social media system.
   * Handles display logic and sends API requests to the backend.
2. **Backend [Spring Boot]**:
   * Manages business logic, processes data, and provides API endpoints.
   * Serves as the central point for handling requests and responses.
3. **WebSocket Service [Spring Boot]**:
   * Supports live chat functionality.
   * Enables communication between users and moderators, logs chat history.
4. **Database [MySQL]**:
   * Stores application data, including user profiles, posts, likes, comments, and chat history.

**Backend API Components**

1. **Authentication Service [Spring Security]**:
   * Handles login credentials, session management, and validation of JWTs (JSON Web Tokens).
   * Ensures secure access to the system.
2. **User Service [Spring Boot]**:
   * Manages user profiles and CRUD operations (Create, Read, Update, Delete).
3. **Post Service [Spring Boot]**:
   * Handles the CRUD operations for posts.
4. **Database Access Layer [JPA/Hibernate]**:
   * Mediates between the backend services and the database.
   * Manages SQL queries and data persistence.

**Frontend Components**

1. **Posts Component [React.js]**:
   * Displays posts and allows users to manage their content.
2. **Authentication Component [React.js]**:
   * Handles login and session management on the client side.
3. **API Service [Axios/Fetch]**:
   * Facilitates communication between the frontend and backend APIs.
   * Sends and retrieves data via HTTP requests.

**CI/CD Pipeline**

1. **Code Repository [GitLab]**:
   * Stores source code and triggers the CI/CD pipeline upon changes.
2. **Build Stage [Gradle]**:
   * Compiles the code and generates build artifacts.
3. **Test Stage [JUnit/Jest]**:
   * Executes unit and integration tests to validate code changes.
4. **SonarQube Stage**:
   * Performs static code analysis to ensure code quality.
5. **Deployment Stage [Docker/Kubernetes]**:
   * Deploys the application to the desired environment.
6. **Developer**:
   * Commits code changes that initiate the pipeline.