APPLICATION DEVELOPMENT

You are tasked with developing a scalable, real-time Collaborative Workspace Platform where users can chat, video call, share files, and collaborate in virtual rooms. The system should support high traffic, ensure scalability, and include core functionalities of real-time interaction and multi-user collaboration.

The project is open-ended, and you can implement additional features as you see fit. The task is divided into three levels, each increasing in complexity. You are encouraged to challenge yourself by tackling higher levels and adding innovative features.

Level 1: Fundamental Features

1. User Authentication System

- Implement signup and sign-in functionality using RESTful APIs.
- Ensure secure password storage (e.g., hashing with bcrypt).

3. Real-Time Chat System

- Develop an instant messaging system within each workspace using WebSockets.
- Support group chats where people can come and leave. Include a feature where the admin can toggle whether newer members can see chat history or not.
- Implement basic features like sending text messages, emojis, and viewing chat history.

4. File Storage and Sharing

- Enable users to upload and download files within a workspace.

- Implement a simple file management system to organize shared files. Consider using s3 or other blob storage tools for the same.

5. Collaborative Document Editing

- Integrate real-time collaborative document editing within workspaces (e.g., using Operational Transforms or CRDTs).
- Support simultaneous editing by multiple users with change tracking.

6. OAuth/SSO Integration

- Integrate third-party authentication providers (e.g., Google, Facebook) for user sign-in/sign-up.
- Implement Single Sign-On (SSO) capabilities for seamless access.
- Consider using nextauth, passport.js, etc.

Additional Level 1 Features:

- **Basic UI/UX Design**: Create a clean and intuitive user interface.

Level 2: Intermediate Features

1. Video Conferencing Integration

- Integrate WebRTC for peer-to-peer video calls within workspaces.
- Support multi-user video conferencing with basic controls (mute, turn off camera).

2. Containerization

- Containerize the application using Docker.

3. Implement load balancer

- Implement load balancing to handle concurrent users (e.g., using Nginx or HAProxy).

4. Caching Mechanism

- Use a caching system like Redis to speed up data retrieval for chat messages and frequently accessed data.
- Implement background service workers to fetch recently used data or pictures and store them in local storage for offline access.

5. Enhanced Security

- Implement role-based access control (RBAC) to manage user permissions within workspaces.

Additional Level 2 Features:

- **Notifications System**: Implement real-time notifications for messages, file uploads, and other events.
    - For websites, use websockets or server-sent events for real-time updates, and Web Push API with service workers for notifications.
    - For apps, use Firebase Cloud Messaging (FCM) for push notifications and real-time updates using Firebase SDK.

Level 3: Advanced Features

1. End-to-End Encryption

- Implement encryption for chat messages and video calls to ensure data privacy.
- Use protocols like Signal Protocol for messaging encryption.

2. Analytics Dashboard

- Use basic ML models to show insights about the data and visualise them using various libraries like matplotlib, or any react packages like shadcn, react-charts, etc.

3. CI/CD Pipeline Integration

- Set up automated deployment pipelines using tools like GitHub Actions or Jenkins.

4. Automated Testing

- Implement unit tests and integration tests and run them automatically for every commit.

6. Advanced Search and Filtering

- Implement advanced search filters and indexing for faster retrieval of information.
- Allow searching across multiple workspaces with permission checks.
- Consider using Elasticsearch for the same.

Additional Level 3 Features:

- **Microservices Architecture**: Refactor the application into microservices for better scalability and maintainability.
- **Integration with Third-Party Services**: Connect with cloud storage providers (e.g., AWS S3, Google Drive) for file storage.

Additional Features to Consider (For All Levels)

- **User Presence Indicators**: Show the online/offline status of users within a workspace.
- **Reaction and Mention Features**: Allow users to react to messages and mention other users.
- **Themes and Customization**: Allow users to customize the appearance of the workspace (e.g., dark mode).
- **Accessibility Features**: Ensure the platform is accessible to users with disabilities (e.g., screen reader support).
- **Logging:** Implement a logging system using tools like Winston or Bunyan. Configure logging levels(eg: error, warn, info) with more verbose logs for local environments and minimal logs for production. Use centralized log manager for scalability and easier debugging.

Guidelines and Expectations

- **Technology Stack**: You are free to choose any programming languages and frameworks. Consider using technologies like Golang, Rust, Node.js, React, or any others you are comfortable with. Either an app or a web is fine.
- **Code Quality**: Write clean, maintainable, and well-documented code. Follow best practices and coding standards.
- **Documentation**: Provide comprehensive documentation, including a README file, API documentation, and setup instructions.

Evaluation Criteria

- **Functionality**: Completion and correctness of the implemented features.
- **Scalability and Performance**: The application's ability to handle high traffic and concurrent users.

- **Innovation**: Creativity in adding new features or improving existing ones.
- **User Experience**: Quality of the UI/UX design and ease of use.
- **Code Quality**: Readability, organization, and adherence to best practices.
- **Documentation and Testing**: Clarity of documentation and thoroughness of testing.

Resources:

- Websocket: https://socket.io/docs/v4/
- Oauth feature: https://www.passportjs.org/docs/ https://next-auth.js.org/getting-started/introduction
- WebRTC: https://webrtc.org/
- Docker: https://docs.docker.com/
- Nginx: https://nginx.org/en/docs/
- Redis: https://redis.io/docs/latest/
- Github Actions: https://docs.github.com/en/actions
- Jenkins: https://www.jenkins.io/doc/
- Youtube resources:
    - https://www.youtube.com/watch?v=zZfhAXfBvVA&list=PLdpzxOOAlwvJDIAQZtMjUUbiVUDfGaCIX
    - https://www.youtube.com/watch?v=Vx2zPMPvmug
    - https://www.youtube.com/watch?v=jgpVdJB2sKQ&t=35s
    - https://www.youtube.com/watch?v=9t9Mp0BGnyI