**Technical Documentation for Video Call KYC Application**

**1. Overview**

**Purpose:**  
A web-based application designed for bank customer KYC (Know Your Customer) processes using video calls. The application manages user queues and facilitates secure video sessions between customers and bank representatives.

**Core Features:**

1. User authentication and registration.
2. Queue management system for video call scheduling.
3. Real-time updates on queue status.
4. Video call functionality powered by Twilio API.
5. Admin dashboard to monitor and manage the queue.

**Target Users:**

1. **Customers:** Individuals seeking to complete KYC via video calls.
2. **Admins:** Bank representatives who process customer queues and conduct video sessions.

**2. Technology Stack**

**Frontend:**

* **React.js with TypeScript:**  
  For building interactive user interfaces and ensuring type safety.
* **Chakra UI:**  
  For a modern and responsive component library.

**Backend:**

* **Java (Spring Boot):**  
  To handle business logic, API endpoints, and token generation for video calls.
* **Twilio SDK:**  
  To manage video call sessions securely.

**Database:**

* **PostgreSQL:**  
  To store user data, queue statuses, and video call session logs.

**Real-Time Communication:**

* **WebSocket or Polling:**  
  To update queue status in real time based on system requirements.

**3. Architecture**

**System Architecture:**  
The application follows a client-server model:

* **Frontend:** Handles UI/UX and interacts with the backend via REST APIs and WebSocket.
* **Backend:** Manages business logic, database interactions, and third-party integrations.
* **Twilio:** Facilitates video call functionality.

**4. Core Functionalities**

**4.1. User Management**

* **Login/Signup:** Secure user authentication using JWT.
* **Profile Management:** Update personal details and track KYC progress.

**4.2. Queue Management**

* Customers are placed in a queue upon login.
* The queue system prioritizes users based on joined\_at timestamp.

**4.3. Video Call**

* **Twilio Video API:** Generates secure tokens for room access.
* Customers and admins join the same room for KYC verification.

**4.4. Real-Time Updates**

* **Polling:** Clients send periodic requests to the server for queue updates.
* **WebSocket:** Push notifications from the server to clients for instant updates.

**4.5. Admin Dashboard**

* View and manage the queue.
* Start or end video calls with users.
* Update user statuses (completed, cancelled).

**5. Database Design**

**Tables**

**1. Users**  
Stores user information.

| **Field** | **Type** | **Description** |
| --- | --- | --- |
| id | serial | Primary key. |
| name | varchar(255) | User's full name. |
| email | varchar(255) | User's email address. |
| password | varchar(255) | Encrypted password. |
| created\_at | timestamp | Registration date. |

**2. Queues**  
Tracks user queue statuses.

| **Field** | **Type** | **Description** |
| --- | --- | --- |
| id | serial | Primary key. |
| user\_id | int | Foreign key referencing users.id. |
| status | varchar(50) | Queue status (waiting, in\_call, completed, cancelled). |
| joined\_at | timestamp | Timestamp when user joined the queue. |
| processed\_at | timestamp | Timestamp when admin started processing. |
| completed\_at | timestamp | Timestamp when the call was completed. |

**6. Workflow**

**6.1. User Workflow**

1. User logs in.
2. Joins the queue and receives a position number.
3. Waits for a video call notification.
4. Completes the call and exits the system.

**6.2. Admin Workflow**

1. Logs into the dashboard.
2. Views the current queue and selects a user to process.
3. Initiates a video call session.
4. Marks the session as completed or cancelled.

**6.3. Video Call Flow**

1. Frontend requests a Twilio token via API.
2. Backend generates the token and returns it.
3. Both parties join the video room using the token.

**7. API Design**

**7.1. Authentication**

**POST /api/auth/login**  
Authenticate user and return a JWT.

**7.2. Queue Management**

**GET /api/queues/status**  
Retrieve the user's position in the queue.

**7.3. Video Call**

**POST /api/calls/token**  
Generate a Twilio token for the video session.

**8. Real-Time Communication**

**Polling:**

* The client periodically requests queue updates (GET /api/queues/status).

**WebSocket:**

* The backend pushes updates to the client when the queue changes:

typescript

Copy code

const socket = new WebSocket("ws://backend-url/queue-updates");

socket.onmessage = (event) => {

console.log("Queue Update:", JSON.parse(event.data));

};

**9. Security**

**Backend Security**

* JWT for user authentication.
* HTTPS for secure communication.

**Video Call Security**

* Twilio tokens with restricted access (room-specific, time-limited).

**Database Security**

* Encrypt sensitive data such as passwords.

**10. Deployment**

**Frontend**

* Build React.js application and host on services like AWS S3, Netlify, or Vercel.

**Backend**

* Deploy Spring Boot service on AWS EC2, Docker containers, or Kubernetes.

**Database**

* Host PostgreSQL on AWS RDS or any cloud database provider.

**11. Tools and Libraries**

**Frontend**

* React.js + TypeScript
* Chakra UI
* Axios (for HTTP requests)

**Backend**

* Spring Boot
* Twilio SDK
* WebSocket Support

**Database**

* PostgreSQL