**LAN-Based Communication System: Backend Development Roadmap**

**Introduction**

This document provides a **detailed roadmap** for developing the backend of a **LAN-based communication system** where students at NIT Silchar can **call, chat, and share files** over a high-speed LAN network without internet dependency. The backend will be built using **Node.js, Express, MongoDB**, and **WebSockets** while leveraging **computer networking principles** to enable seamless data exchange.

**Project Scope and Features**

**Authentication & Authorization**

* **Login Mechanism:**
  + Upon connecting to LAN, a login page will automatically open.
  + Students will enter a **unique identifier (Scholar ID)**.
  + An **OTP** will be generated and displayed on the screen.
  + Upon **OTP verification**, access is granted.
* **Security Measures:**
  + **JWT-based authentication** for secure session management.
  + **MAC Address & IP Binding** for device authentication.

**Chat & Calling Features**

* **Real-time messaging** over LAN using **Socket.io**.
* **Voice and video calls** over LAN (WebRTC integration).
* **Group chats and direct messaging**.

**File Sharing System**

* **Peer-to-Peer (P2P) file transfer**.
* **High-speed local file exchange** with WebRTC DataChannel.

**Technology Stack**

**Backend Technologies**

* **Node.js** → Runtime for backend logic.
* **Express.js** → Web framework for API handling.
* **MongoDB** → NoSQL database for storing users, messages, and files.
* **Socket.io** → Enables real-time messaging and notifications.
* **WebRTC** → Handles peer-to-peer calling and file sharing.

**Networking Concepts Used**

* **LAN-based IP Addressing** → Devices communicate using assigned local IPs (e.g., 192.168.x.x).
* **MAC Address Authentication** → Ensures identity verification of LAN users.
* **DHCP & DNS Handling** → Captive portal setup for login redirection.
* **WebRTC Peer-to-Peer** → Establishes direct media and data transfer.
* **Network Ports and Sockets** → Socket.io for chat, WebRTC for calls.
* **Multicast & Broadcast Messaging** → Efficient LAN-wide announcements.

**Backend Development Roadmap**

**Step 1: Project Setup**

**Initialize Project**

1. Create project directory: mkdir lan-auth-system && cd lan-auth-system.
2. Initialize Node.js: npm init -y.
3. Install dependencies:
4. npm install express mongoose dotenv cors body-parser jsonwebtoken socket.io macaddress

**Project Structure**

lan-auth-system/

│── models/ # Database Schemas

│── routes/ # API Endpoints

│── config/ # Server & Database Configuration

│── middleware/ # Authentication & Security

│── server.js # Main Entry Point

│── package.json # Dependencies

│── .env # Environment Variables

**Step 2: Database Setup (MongoDB)**

**Schema Design**

* **Users Schema (User.js)**
* const userSchema = new mongoose.Schema({
* scholarId: { type: String, required: true, unique: true },
* macAddress: { type: String, required: true, unique: true },
* ipAddress: { type: String, required: true },
* isVerified: { type: Boolean, default: false },
* lastLogin: { type: Date, default: Date.now }
* });
* **OTP Schema (OTP.js)**
* const otpSchema = new mongoose.Schema({
* scholarId: { type: String, required: true },
* otp: { type: String, required: true },
* createdAt: { type: Date, default: Date.now, expires: 300 }
* });

**Step 3: Authentication System**

**Generate OTP API (auth.js)**

* Generates OTP, stores user MAC & IP details.

**Verify OTP API (auth.js)**

* Validates OTP and provides JWT session.

**Logout API (auth.js)**

* Ends session by marking user unverified.

**Step 4: Real-time Chat & Calls**

**Using WebSockets (server.js)**

* Establish Socket.io server.
* Handle real-time messages & presence status.
* Implement WebRTC for voice/video calls.

**Step 5: File Sharing System**

**P2P File Sharing (WebRTC)**

* Direct file transfers over LAN.
* No need for a central server.
* Faster than traditional HTTP file transfers.

**Step 6: Network Optimization**

**Load Balancing & Scalability**

* Optimize WebSockets for multiple concurrent users.
* Implement multicast messages for efficient LAN-wide communication.

**Security Measures**

* **Encryption of messages**.
* **Token-based authentication**.
* **Restricting unauthorized MAC addresses**.

**Frontend Considerations**

* **Login Page:** Auto-opens on LAN connection.
* **Chat UI:** Interactive message interface.
* **Calling UI:** WebRTC-based call buttons.
* **File Sharing UI:** Drag & Drop file upload.

**Conclusion**

This roadmap provides a **detailed breakdown** of the backend implementation and **networking principles** behind a **LAN-based communication system**. With the backend in place, the **frontend team** can now focus on UI/UX design while ensuring smooth integration with the APIs and WebSockets provided.

**Next Steps**

* **Develop and integrate frontend.**
* **Optimize performance for large user load.**
* **Test for various LAN configurations.**

🚀 **This system will revolutionize communication within the NIT Silchar campus by enabling high-speed, secure, and real-time data exchange over LAN!**