ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH

**TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN**

****

**BÁO CÁO TỔNG HỢP**

**KẾT QUẢ THỰC HIỆN**

**ĐỒ ÁN AN TOÀN THÔNG TIN – Tuần 4**

**Nội dung: Thiết kế và xây dựng hệ thống**

**Tên đồ án: KumoChat – Hệ thống chat nội bộ**

**Giáo viên hướng dẫn: Nguyễn Đình Thúc**

**Sinh viên thực hiện: Trần Bùi Thái Sơn**

**Tp. Hồ Chí Minh, tháng …… năm ……**

**Giai đoạn 1: Lên kế hoạch**

**1.1. Xác định tính năng chính**

**A diagram of a group chat

AI-generated content may be incorrect.**

**1.2. Mô hình mã hóa**

* Private chat:
  + Chat riêng tư giữa các thành viên ở trong một phòng ban
  + E2EE: mỗi tin nhắn gửi đi được mã hóa bởi một AES Key bất kỳ, AES Key sau đó được mã hõa bởi Public Key người nhận và người gửi, người nhận và người gửi sau đó đều có thể đọc lại bằng cách giải mã AES Key bằng Private Key, giải mã tin nhắn bằng AES Key.
  + Server chỉ nhận và gửi tin nhắn đã được mã hóa, tin nhắn được lưu xuống Database đều ở dạng mã hóa.
  + Đảm bảo sự bí mật cho cuộc trò chuyện.
* Group chat:
  + Nhóm chat của một phòng ban, thông báo các vấn đề cần thiết ở trong một phòng ban.
  + E2EE: Mỗi group chat có một AES Key được tạo sẵn, AES Key này ban đầu được mã hóa bởi Public Key của Admin. Khi User được thêm vào group chat, Admin giải mã AES Key bằng Private Key của mình, sau đó mã hóa AES Key group chat bằng Public Key của User được thêm vào. Hoàn thành công việc chia sẻ AES Key group chat.
* Broadcast chat:
  + Khi Admin cần thông báo đến toàn bộ User. User chỉ nhận tin broadcast, không thể gửi.
  + E2EE: Mỗi tin broadcast được mã hóa bởi một AES Key bất kì và gửi tin đến tất cả mọi người, AES Key được mã hóa bởi Public Key của từng User.

**1.3. Chọn công nghệ**

* **Backend:** Node.js + Express + PostgreSQL, ClamAV, VirusTotal API
  + ClamAV, VirusTotal API: Kiểm tra file upload có mã độc không trước khi gửi hay lưu vào database.
  + Node.js: Xử lý bất đồng bộ tốt, thích hợp cho các ứng dụng real-time như chat.
  + Express: Nhẹ, linh hoạt, dễ cấu hình các middleware bảo mật như rate-limiting, helmet, CORS.
  + PostgreSQL: Cơ sở dữ liệu quan hệ mạnh mẽ, hỗ trợ kiểm soát truy cập chặt chẽ và mã hóa ở mức cơ sở dữ liệu (TDE, SSL).
* **Auth:** JWT
  + Token có thể ký bằng HMAC hoặc RSA, khó giả mạo.
  + Có thể đính thêm thông tin như role, thời hạn sống (exp), v.v.
  + Cho phép xác thực stateless, lý tưởng cho ứng dụng phân tán.
* **File storage:** Local disk
  + Lưu trữ nội bộ, đơn giản và kiểm soát tốt truy cập.
  + Ít phụ thuộc bên thứ ba (không cần cloud...).
* **Real-time:** Socket.io
  + Hỗ trợ xác thực kết nối thông qua JWT hoặc token tùy chỉnh.
  + Có thể mã hóa dữ liệu gửi nhận ở tầng ứng dụng.
* **Frontend:** React
  + Component-based, dễ kiểm soát từng phần UI.
  + Dễ tích hợp các biện pháp như XSS protection (qua JSX auto-escape), CSRF token.
* **ORM:** Prisma
  + Tránh SQL Injection nhờ query builder an toàn.
  + Hỗ trợ schema migration có kiểm soát, dễ rollback nếu phát hiện lỗi.
* **Deployment:** Backend = Render, Frontend = Vercel
  + Render: Tích hợp HTTPS tự động, firewall cơ bản, environment variables được bảo mật
  + Vercel: Cũng hỗ trợ HTTPS, headers bảo mật mặc định, dễ thiết lập chính sách CORS an toàn.

|  |  |
| --- | --- |
|  |  |

**Giai đoạn 2: Thiết kế Database & Backend**

**2.1. Thiết kế cơ sở dữ liệu**

* Tạo các bảng: Users, Departments, User\_Departments, Messages, Login\_Logs,…

**2.2. Viết các API cần thiết**

**a. Auth APIs**

| **Endpoint** | **Method** | **Mô tả** |
| --- | --- | --- |
| /auth/register | POST | Đăng ký user, client gửi username, password, publicKey, encryptedPrivateKey |
| /auth/login | POST | Đăng nhập, trả JWT token |
| /auth/me | GET | Lấy thông tin cá nhân |

**b. Admin APIs**

| **Endpoint** | **Method** | **Mô tả** |
| --- | --- | --- |
| /admin/approve-user/:id | POST | Duyệt user đăng ký |
| /admin/create-department | POST | Tạo nhóm/phòng ban mới |
| /admin/add-user-to-department | POST | Thêm user vào nhóm, đồng thời mã hóa group key bằng publicKey và lưu vào GroupKeyShare |
| /admin/assign-leader | POST | Gán 1 user làm leader của 1 nhóm |
| /admin/send-broadcast | POST | Gửi tin nhắn broadcast đến toàn công ty |
| /admin/users | GET | Danh sách user (để phân quyền, duyệt, gán nhóm) |
| /admin/logs | GET | Xem log login / logout |

**c. Messaging APIs**

| **Endpoint** | **Method** | **Mô tả** |
| --- | --- | --- |
| /messages/private | POST | Gửi tin nhắn riêng tư (gửi ciphertext + AES key mã hóa RSA) |
| /messages/group | POST | Gửi tin nhóm (gửi ciphertext, sử dụng shared AES key đã giải mã từ trước) |
| /messages/broadcast | GET | Lấy tin nhắn broadcast (tất cả user đều có thể xem) |
| /messages/group/:departmentId | GET | Lấy tin nhắn trong nhóm cụ thể |
| /messages/private/:userId | GET | Lấy tin nhắn 1-1 giữa bạn và user khác |

**d. Key Management APIs**

| **Endpoint** | **Method** | **Mô tả** |
| --- | --- | --- |
| /keys/my | GET | Lấy publicKey và encryptedPrivateKey của chính mình |
| /keys/public/:userId | GET | Lấy publicKey của người khác (để mã hóa AES key khi gửi private) |
| /keys/group/:departmentId | GET | Lấy group key đã mã hóa RSA cho chính user đó (dùng khi vào nhóm mới) |

**e. File Upload APIs**

| **Endpoint** | **Method** | **Mô tả** |
| --- | --- | --- |
| /files/upload | POST | Upload file đính kèm |
| /files/:id | GET | Tải file về nếu có quyền |

**Giai đoạn 3: Authentication & Authorization**

* Hash password bằng bcrypt
* Xác thực JWT cho tất cả các route
* Middleware isAdmin, isLeader, isGroupMember để bảo vệ các endpoint
* Ghi chép login/logout vào Login\_Logs

**Giai đoạn 4: Messaging + Socket.io (Realtime)**

* Thiết lập Socket.io
* Khi user login → join room theo userId
* Group chat → join theo departmentId
* Broadcast → emit tới tất cả client
* Lưu tin nhắn (kể cả file) vào bảng Messages

**Giai đoạn 5: Frontend - Thiết kế các Component cần có:**

* Đăng nhập / Đăng ký
* Chat (private / group / broadcast)
* Quản lý nhóm (admin)
* Upload file, kiểm tra file có mã độc không và lưu file
* Notification & trạng thái online

**Giai đoạn 6: Testing & Deployment**

* Unit test cho API (dùng Jest hoặc Mocha)
* Test chat realtime với nhiều account
* Deploy Backend (Render)
* Deploy Frontend (Vercel)
* Tạo .env để cấu hình biến môi trường