# System Design Document – Chat It

## 1. Introduction

### 1.1 Purpose

This System Design Document describes the overall architecture, components, data flow, and interaction between modules of the **Chat It** application. It acts as a technical blueprint for implementing the system based on the approved SRS.

### 1.2 Scope

The document covers: - High-level and low-level system architecture - Component design - Database design overview - Real-time communication flow - Security and scalability considerations

---

## 2. System Architecture

### 2.1 High-Level Architecture

Chat It follows a **client–cloud architecture** using Backend-as-a-Service (BaaS).

**Main Layers:** 1. Presentation Layer (Mobile App) 2. Application Services Layer (Supabase) 3. External Services Layer (Music APIs)

```
User
  ↓
React Native App (Expo)
  ↓
Supabase SDK
  ├── Authentication
  ├── PostgreSQL Database
  ├── Realtime Engine
  └── Storage
  ↓
External APIs (YouTube / SoundCloud)
```

---

# 3. Component Design

## 3.1 Client Side Components (React Native)

- **Authentication Module**
- Login / Signup
- Token management

- **Profile Module**

- View/Edit profile
- Follow/Unfollow users

- **Feed Module**

- Create posts
- View feed
- Like & comment

- **Story Module**

- Upload stories
- View stories

- **Chat Module**

- One-to-one chat
- Media messages
- Duo room

- **Music Sync Module (Tune Together)**

- Start music session
- Sync play/pause

- **Mini Games Module (Play It)**

- Game UI
- Turn-based logic

---

### 3.2 Backend Components (Supabase)

- **Supabase Auth**
- User authentication
- Session handling
- **PostgreSQL Database**
- Stores users, posts, chats, messages, games
- **Realtime Engine**
- Live chat updates
- Game state sync
- Music session sync
- **Supabase Storage**
- Media files (images, videos, audio)

---

## 4. Database Design Overview

### 4.1 Core Tables

- users
- posts
- followers
- stories
- chats
- messages
- music_sessions
- games

### 4.2 Relationships

- One user → many posts
- One chat → many messages
- One chat → one active music session
- One chat → one active game session

---

# 5. Data Flow Design

### 5.1 User Login Flow

1. User enters credentials
2. Request sent to Supabase Auth
3. Token returned
4. User session created in app

---

### 5.2 Chat Message Flow

1. User sends message
2. Message stored in `messages` table
3. Supabase Realtime broadcasts update
4. Receiver gets message instantly

---

### 5.3 Music Sync Flow

1. User starts music session
2. Song ID stored in `music_sessions`
3. Play/Pause updates sent via Realtime
4. Both users stay synchronized

---

### 5.4 Mini Game Flow

1. User starts a game
2. Game state stored in `games` table
3. Each move updates state
4. Realtime sync updates opponent UI

---

# 6. Security Design

- Row Level Security (RLS) on all tables
- Only chat participants can read messages
- Users can modify only their own data
- Secure JWT-based authentication

---

# 7. Scalability & Performance

- Supabase handles horizontal scaling

- Realtime channels limited per chat
- Pagination for feed and messages
- Media delivery via CDN

---

## 8. Error Handling & Logging

- Client-side validation
- Graceful API error handling
- Supabase logs for backend monitoring

---

## 9. Deployment Design

- Mobile app built using Expo
- Backend hosted on Supabase cloud
- CI/CD via Expo build service

---

## 10. Future Enhancements

- Group chats with shared music
- Multiplayer games
- Recommendation engine
- Admin moderation dashboard

---

## 11. Conclusion

This system design ensures a scalable, secure, and real-time capable architecture for Chat It, supporting both traditional social media features and innovative shared experiences.

---

**Document Version:** 1.0
**Status:** Approved for Implementation