# Unity Chat App – Design Document

## 1. Overview

A cross-platform chat application (mobile & desktop) inspired by Discord and Snapchat. Users can chat with friends, see online status, and customize their settings. Data is stored in **Spacetime DB** and authentication is handled via **Auth0 (OpenID)**.

## 2. Platforms

* **Mobile:** iOS & Android
* **Desktop:** Windows & macOS
* **UI/UX:** Dark theme, Discord-inspired

## 3. Authentication

* **Service:** Auth0
* **Method:** OpenID Connect (OIDC)
* **Flow:**
  1. User logs in via Auth0.
  2. Receive ID token / access token.
  3. Token is used to identify the user in Spacetime DB (ctx.Sender).

## 4. Data Model (Spacetime DB)

### Tables

#### User

Identity Identity; // Primary Key  
bool Online;  
UserSettings Settings; // Name, Color, ConnectSoundID, DisconnectSoundID

#### Message

Identity Sender;  
Timestamp Sent;  
string ChannelID; // "" = main channel, shared key = DM  
string Text;

### UserSettings

string Name;  
string Color;  
int ConnectSoundID;  
int DisconnectSoundID;

## 5. Core Features

1. **User Presence**
   * Online/offline status displayed next to username.
   * Updated automatically on login/logout.
2. **Chat**
   * **Main channel:** channelID = ""
   * **DMs:** channelID = shared key (TBD key generation)
   * Messages displayed in chronological order with sender info.
3. **Messaging**
   * Input field at bottom.
   * Send messages via SendMessage reducer.
   * Prevent empty messages.
4. **User Settings**
   * Update via SetUserSettings reducer.
   * Customizable name, color, connect/disconnect sounds.

## 6. UI/UX Layout

| Section | Position | Notes |
| --- | --- | --- |
| Online users list | Left / Side panel | Show online/offline indicator |
| Chat messages | Center | Scrollable, newest at bottom |
| Input field | Bottom | Text input + send button |
| User settings | Top-right / Modal | Color, name, sounds |

* Dark theme (Discord-inspired)
* Responsive layout for desktop & mobile

## 7. Components & Scripts (Unity)

* **UIManager:** Handles layout and updates UI based on DB events.
* **AuthManager:** Handles Auth0 login flow and token management.
* **ChatManager:** Handles sending/receiving messages via Spacetime DB.
* **UserManager:** Tracks user status and settings.
* **DMManager:** Handles DM creation and private channel keys (TBD).
* **SoundManager:** Plays connect/disconnect sounds based on UserSettings.

## 8. Data Flow

1. User logs in → Auth0 token → Unity → Spacetime DB identifies user.
2. Online status set → DB updates User.Online.
3. User sends message → SendMessage reducer → DB inserts into Message table.
4. DB triggers update → UIManager refreshes message list.
5. User updates settings → SetUserSettings reducer → DB updates UserSettings.

## 9. Messaging Structure

* **Main Channel:** channelID = "" (public chat)
* **Direct Messages:** channelID = <shared key>
  + Shared key could be a deterministic combination of user IDs.
  + Only users in the DM have access.