# Unity Chat App – Design Document V2

## 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)**.

## Platforms

* **Mobile:** Android
* **Desktop:** Windows

## Authentication

* **Service:** Auth0
* **Method:** OpenID Connect (OIDC) (Required by SpacetimeDB)
* **Flow:**
  + User logs in via Auth0. (Unity Opens Browser window)
  + Receive ID token / access token (Token will be saved locally).
  + Token is used to identify the user in SpacetimeDB.

## Data Model (Spacetime DB)

### User

* Identity Identity;
* Bool Online;
* UserSettings Settings;

### Message

* Identity Sender;
* Timestamp Sent;
* String ChannelID;
* String Text;

### UserSettings

* String Name;
* String Color;
* Int ConnectSoundID;
* Int DisconnectSoundID;

## Core Features

1. **User Presence**
   * Online/offline status displayed next to username.
   * Updated automatically on login/logout.
2. **Chat**
   * Main channel
   * DMs?(TBD, data is there, but no plan to implement yet).
   * Messages displayed in chronological order with sender info.
3. **Messaging**
   * Input field at bottom.
   * Send messages.
   * Prevent empty messages.
4. **User Settings**
   * Update user Settings.
   * Customizable name, color, connect/disconnect sounds.

## 6. UI/UX Layout

* Dark theme (Discord-inspired)
* Responsive layout for desktop & mobile
* Areas for:
  + Online list
  + Text input
  + Message list
  + DM list (TBD)

## 7. Components & Scripts (Unity)

* **ChatManager:** Handles sending/receiving messages via Spacetime DB.
* **UserListManager:** Tracks user status and settings.
* **AudioManager:** Plays connect/disconnect sounds based on UserSettings.
* **NetworkManager:** handles the connection with the server, also keeps a reference to the server.
* **InputManager:** handles user input back to the server, queues input to avoid race conditions.
* **MessageHandler:** handles taking the users message input and enqueues it with the input manager.
* **UserSettingsMenu:** handles the menu where you will be able to change user settings, which will be enqueued with the input manager.

## 8. Data Flow

* User logs in → Auth0 token → Unity → Spacetime DB identifies user.
* Online status set → DB updates User.Online.
* User sends message → SendMessage reducer → DB inserts into Message table.
* DB triggers update → Multiple scripts listen to events to update their values
* User updates settings → SetUserSettings reducer → DB updates UserSettings.