Fake Discord Application Database

Design Document

INFORMATION MANAGEMENT FINAL PROJECT

Last Updated: March 06, 2025

Prepared by:

ESPIN, KYLE CEDRIC

CADIZ, ANTONETTE MHYLS

1. Introduction

Project Name: FakeDiscordApp

Purpose: The database is designed to support a Discord-like application, enabling users to communicate via direct messages and channels, join servers, and manage friendships.

Scope: The database will store user information, friendships, direct messages, servers, channels, channel messages, server members, and server invites.

Audience: Developers, Database Administrators (DBAs), and stakeholders.

Version: 1.0

Last Updated: 3/6/2025

---

2. Database Overview

Database Type: Relational Database

**Database Management System (DBMS): MySQL**

Database Name: fake_discord

Description: The database is designed to manage user interactions, server memberships, and messaging functionalities for a Discord-like application.

3. Entity-Relationship Diagram (ERD)

Below is a high-level description of the ERD:

- **Users can send Direct Messages to other users.**

- **Users can have Friends (many-to-many relationship).**

- **Servers are owned by a user and contain multiple Channels.**

- **Channels contain Channel Messages sent by users.**

- **Server Members represent users who are part of a server.**

- **Server Invites allow users to invite others to join a server.**
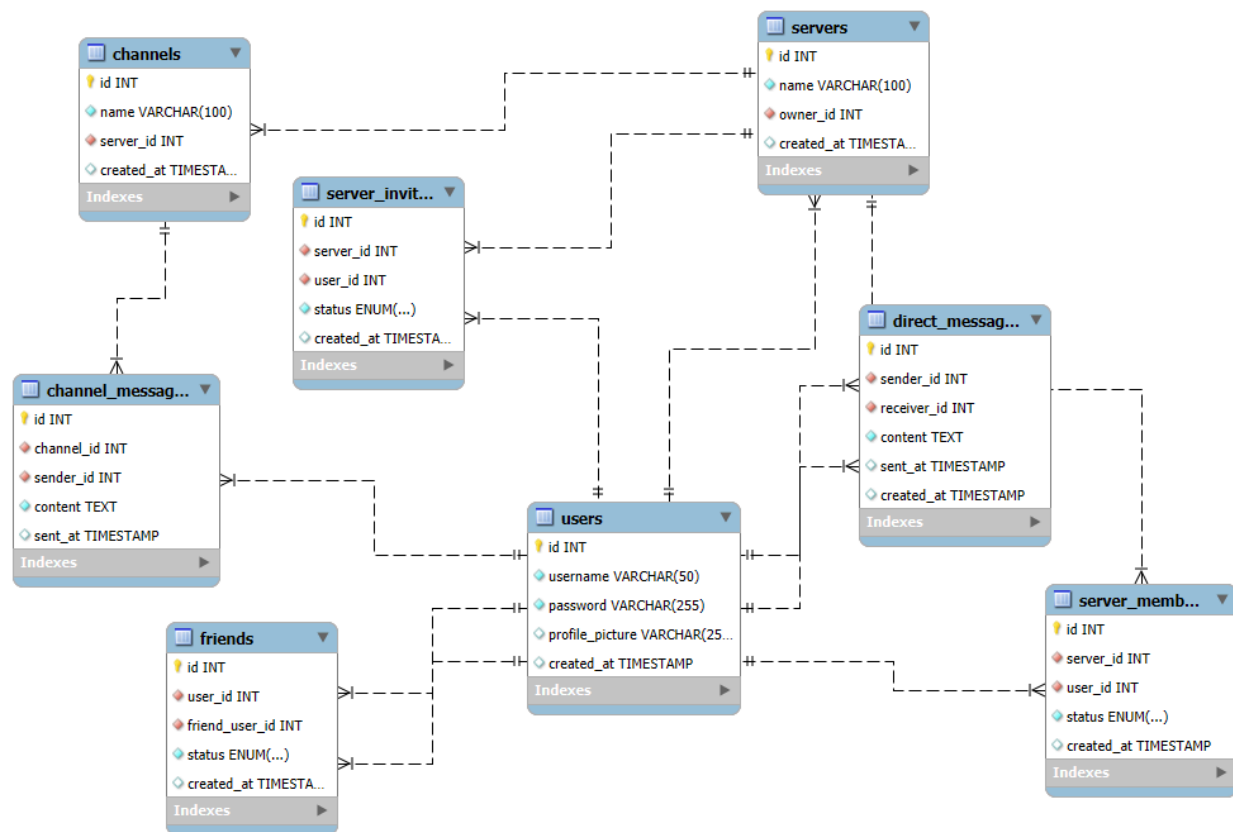
4. Tables and Schema

Table Name: users

Description: Stores user information.

Column Name	Data Type	Constraints	Description
id	INT	PRIMARY KEY, AUTO_INCREMENT	Unique identifier for the user
username	VARCHAR(50)	NOT NULL, UNIQUE	User's username
password	VARCHAR(255)	NOT NULL	Hashed password
profile_picture	VARCHAR(255)		Path to the user's profile picture
created_at	TIMESTAMP	DEFAULT CURRENT_TIMESTAMP	Timestamp of user creation

Table Name: friends

Description: Stores friend relationships between users.

Column Name	Data Type	Constraints	Description
id	INT	PRIMARY KEY, AUTO_INCREMENT	Unique identifier for the friendship
user_id	INT	NOT NULL	ID of the user
friend_user_id	INT	NOT NULL	ID of the friend
status	ENUM	NOT NULL, DEFAULT 'pending'	Status of the friendship
created_at	TIMESTAMP	DEFAULT CURRENT_TIMESTAMP	Timestamp of friendship creation

Foreign Keys:

- `user_id` references `users(id)` ON DELETE CASCADE

- `friend_user_id` references `users(id)` ON DELETE CASCADE

Unique Constraint:

- `UNIQUE(user_id, friend_user_id)`

Table Name: direct_messages

Description: Stores private messages between users.

Column Name	Data Type	Constraints	Description
id	INT	PRIMARY KEY, AUTO_INCREMENT	Unique identifier for the message
sender_id	INT	NOT NULL	ID of the sender
receiver_id	INT	NOT NULL	ID of the receiver
content	TEXT	NOT NULL	Content of the message
sent_at	TIMESTAMP	DEFAULT CURRENT_TIMESTAMP	Timestamp of message sending

Foreign Keys:

- **sender_id** references **users(id)** ON DELETE CASCADE

- **receiver_id** references **users(id)** ON DELETE CASCADE

Table Name: servers

Description: Stores server information.

Column Name	Data Type	Constraints	Description
id	INT	PRIMARY KEY, AUTO_INCREMENT	Unique identifier for the server
name	VARCHAR(100)	NOT NULL	Name of the server
owner_id	INT	NOT NULL	ID of the server owner
created_at	TIMESTAMP	DEFAULT CURRENT_TIMESTAMP	Timestamp of server creation

Foreign Key:

- `owner_id` references `users(id)` ON DELETE CASCADE

Table Name: channels

Description: Stores channels within servers.

Column Name	Data Type	Constraints	Description

id	INT	PRIMARY KEY, AUTO_INCREMENT	Unique identifier for the channel
name	VARCHAR(100)	NOT NULL	Name of the channel
server_id	INT	NOT NULL	ID of the server
created_at	TIMESTAMP	DEFAULT CURRENT_TIMESTAMP	Timestamp of channel creation

Foreign Key:

- `server_id` references `servers(id)` ON DELETE CASCADE

---

5. TABLE PURPOSES

Users Table:

- **Stores user information (username, password, profile picture).**
- **Tracks account creation date.**

Friends Table:

- **Manages friend relationships between users.**
- **Supports friendship statuses (e.g., pending, accepted).**

Channel Messages Table:

- **Stores messages sent within server channels.**

- **Records message content and sender details.**

Servers Table:

- **Holds information about servers (name, owner).**

- **Links each server to its creator.**

Direct Messages Table:

- **Stores private messages between users.**

- **Tracks when messages are sent.**

Server Members Table:

- **Manages users who are part of a server.**

- **Tracks membership status (pending, accepted).**

Channels Table:

- **Stores channels within each server.**

- **Links channels to their respective servers.**

Server Invites Table:

- **Stores invitations to join a server.**

- **Records invitation status and related details.**

5. Relationships

- **Users and Friends: Many-to-Many (via `friends` table).**

- **Users and Direct Messages: One-to-Many (a user can send/receive many messages).**

- **Servers and Channels: One-to-Many (a server can have many channels).**

- **Channels and Channel Messages: One-to-Many (a channel can have many messages).**

- **Servers and Server Members: Many-to-Many (via `server_members` table).**

- **Servers and Server Invites: One-to-Many (a server can have many invites).**

---

6. Constraints

- **Primary Keys: Each table has a primary key (`id`).**

- **Foreign Keys: All foreign keys are defined to maintain referential integrity.**

- **Unique Constraints:**

  - **`users(username)` ensures unique usernames.**

  - **`friends(user_id, friend_user_id)` ensures unique friendships.**

- **Check Constraints: None explicitly defined.**

7. Security Considerations

- **Access Control: Only authorized users (application backend) can access the database.**

- **Encryption: Passwords are stored as hashed values.**

- **Backup Strategy: Regular backups will be performed (e.g., daily).**

- **Audit Logging: Logs will track critical operations (e.g., user deletions).**

---

8. Backup and Recovery

- **Backup Frequency: Daily backups.**

- **Backup Location: Cloud storage or secure on-premise storage.**

- **Recovery Process: Restore from the latest backup and replay transaction logs.**

---

9. Appendices

- **Glossary:**

  - **DBMS: Database Management System.**

  - **ERD: Entity-Relationship Diagram.**

- **References: MySQL Documentation.**