Team J

Milestone 2: Specifications

1. Entities and Attributes

List each table and its key fields.

At minimum, include tables for:

- Users — login info, display name, etc.
- Games — game type, status, timestamps, etc.
- Game Participants — records of which users are in which games
- Additional tables as needed (e.g., scores, chat messages, settings)

Each table should identify:

- Primary key (PK)
- Foreign keys (FK)
- Column names and types (e.g., VARCHAR, INT, BOOLEAN, etc.)

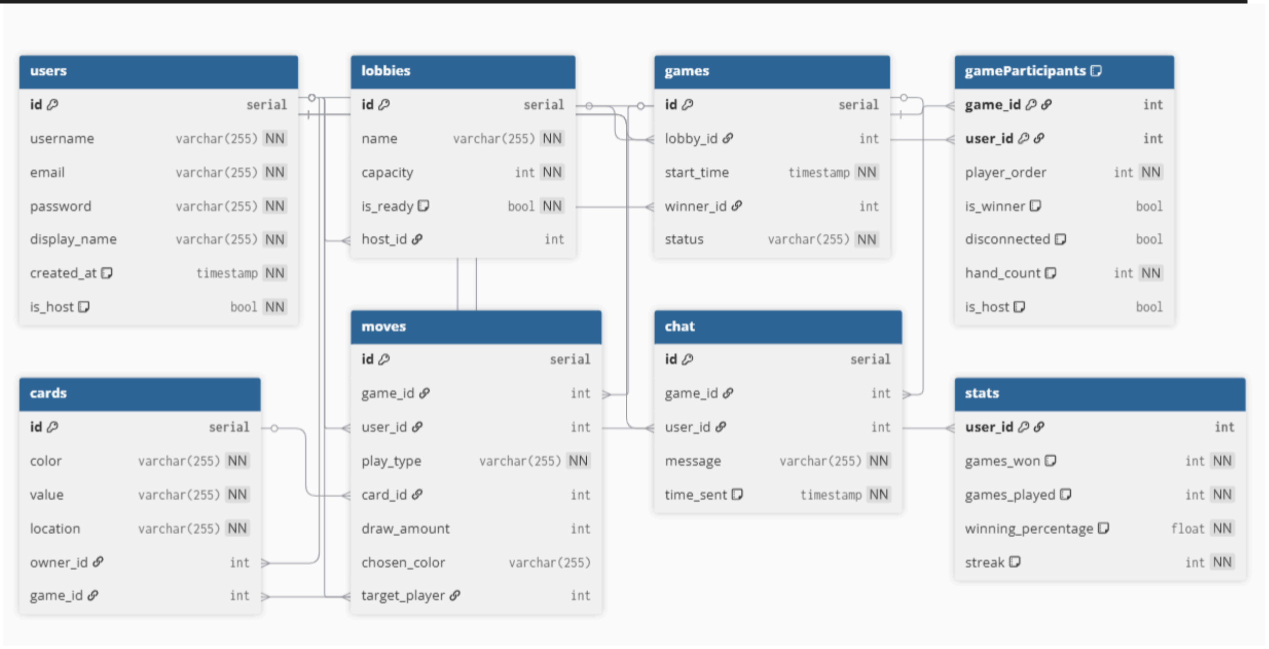
User ● id — SERIAL — Primary Key ● username — VARCHAR(255) ● email — VARCHAR(255) ● password — VARCHAR(255) ● display_name — VARCHAR(255) ● created_at — TIMESTAMP ● is_host — BOOLEAN	**Game Participants** ● game_id — INT — Foreign Key references games(id) ● user_id — INT — Foreign Key reference user(id) ● player_order — INT ● is_winner — BOOLEAN ● disconnected — BOOLEAN ● hand_count — INT ● is_host — BOOLEAN ● Primary Key — composite (game_id, user_id)
Lobby ● id — SERIAL — Primary Key ● name — VARCHAR(255) ● capacity — INT ● is_ready — BOOLEAN ● host_id — INT — Foreign Key reference users(id)	**Game** ● id — SERIAL — Primary Key ● lobby_id — INT — Foreign Key references lobbies(id) ● start_time — TIMESTAMP ● winner_id — INT — Foreign Key references user(id) ● status — VARCHAR(255)
Stats ● user_id — INT — Primary Key, Foreign Key references users(id) ● games_won — INT	**Chat** ● id — SERIAL — Primary Key

● games_played — INT ● winning_percentage — FLOAT ● streak — INT	● game_id — INT — Foreign Key references games(id) ● user_id — INT — Foreign Key references users(id) ● message — VARCHAR(255) ● time_sent — TIMESTAMP — default CURRENT_TIMESTAMP
Move ● id — SERIAL — Primary Key ● game_id — INT — Foreign Key references games(id) ● user_id — INT — Foreign Key references users(id) ● play_type — VARCHAR(255) ● card_id — INT — Foreign Key references cards(id) ● draw_amount — INT ● chosen_color — VARCHAR(255) ● target_player — INT — Foreign Key references users(id)	**Card** ● id — SERIAL — Primary Key ● color — VARCHAR(255) ● value — VARCHAR(255) ● location — VARCHAR(255) ● owner_id — INT — Foreign Key reference users(id) ● game_id — INT — Foreign Key references games(id)

dbdiagram.io code and screenshot

```
Table users {
  id             serial [pk, unique]
  username       varchar(255) [not null, unique]
  email          varchar(255) [not null, unique]
  password       varchar(255) [not null]
  display_name   varchar(255) [not null]
  created_at     timestamp [not null, default: `CURRENT_TIMESTAMP`]
  is_host        bool [not null, default: false]
}

Table lobbies {
  id         serial [pk, unique]
  name       varchar(255) [not null]
  capacity   int [not null]
  is_ready   bool [not null, default: false]
  host_id    int           // REFERENCES users(id) ON DELETE CASCADE
}

Table games {
  id          serial [pk, unique]
  lobby_id    int          // REFERENCES lobbies(id) ON DELETE CASCADE
  start_time timestamp [not null]
  winner_id   int          // REFERENCES users(id) ON DELETE CASCADE
  status      varchar(255) [not null]
}

Table gameParticipants {
  game_id        int       // REFERENCES games(id) ON DELETE CASCADE
  user_id        int       // REFERENCES users(id) ON DELETE CASCADE
  player_order   int [not null]
  is_winner      bool [default: false]
  disconnected   bool [default: false]
  hand_count     int  [not null, default: 0]
  is_host        bool [default: false]

  indexes {
    (game_id, user_id) [pk]
  }
}

Table cards {
  id         serial [pk, unique]
  color      varchar(255) [not null]
  value      varchar(255) [not null]
  location   varchar(255) [not null]
  owner_id   int          // REFERENCES users(id) ON DELETE CASCADE
  game_id    int          // REFERENCES games(id) ON DELETE CASCADE
}

Table moves {
  id            serial [pk, unique]
  game_id       int          // REFERENCES games(id) ON DELETE CASCADE
  user_id       int          // REFERENCES users(id) ON DELETE CASCADE
  play_type     varchar(255) [not null]
  card_id       int          // REFERENCES cards(id) ON DELETE CASCADE
  draw_amount   int
  chosen_color  varchar(255)
  target_player int          // REFERENCES users(id) ON DELETE CASCADE
}

Table chat {
  id         serial [pk, unique]
  game_id    int          // REFERENCES games(id) ON DELETE CASCADE
  user_id    int          // REFERENCES users(id) ON DELETE CASCADE
  message    varchar(255) [not null]
  time_sent  timestamp [not null, default: `CURRENT_TIMESTAMP`]
}

Table stats {
  user_id              int [pk] // REFERENCES users(id) ON DELETE CASCADE
  games_won            int [not null, default: 0]
  games_played         int [not null, default: 0]
  winning_percentage   float [not null, default: 0]
  streak               int [not null, default: 0]
}
```

```
// Foreign keys
Ref: lobbies.host_id > users.id [delete: cascade]
Ref: games.lobby_id > lobbies.id [delete: cascade]
Ref: games.winner_id > users.id [delete: cascade]
Ref: gameParticipants.game_id > games.id [delete: cascade]
Ref: gameParticipants.user_id > users.id [delete: cascade]
Ref: cards.game_id > games.id [delete: cascade]
Ref: cards.owner_id > users.id [delete: cascade]
Ref: moves.game_id > games.id [delete: cascade]
Ref: moves.user_id > users.id [delete: cascade]
Ref: moves.card_id > cards.id [delete: cascade]
Ref: moves.target_player > users.id [delete: cascade]
Ref: chat.game_id > games.id [delete: cascade]
Ref: chat.user_id > users.id [delete: cascade]
Ref: stats.user_id > users.id [delete: cascade]
```



users		
id	serial	
username	varchar(255)	NN
email	varchar(255)	NN
password	varchar(255)	NN
display_name	varchar(255)	NN
created_at	timestamp	NN
is_host	bool	NN

lobbies		
id	serial	
name	varchar(255)	NN
capacity	int	NN
is_ready	bool	NN
host_id	int	

games		
id	serial	
lobby_id	int	
start_time	timestamp	NN
winner_id	int	
status	varchar(255)	NN

gameParticipants		
game_id	int	
user_id	int	
player_order	int	NN
is_winner	bool	
disconnected	bool	
hand_count	int	NN
is_host	bool	

cards		
id	serial	
color	varchar(255)	NN
value	varchar(255)	NN
location	varchar(255)	NN
owner_id	int	
game_id	int	

moves		
id	serial	
game_id	int	
user_id	int	
play_type	varchar(255)	NN
card_id	int	
draw_amount	int	
chosen_color	varchar(255)	
target_player	int	

chat		
id	serial	
game_id	int	
user_id	int	
message	varchar(255)	NN
time_sent	timestamp	NN

stats		
user_id	int	
games_won	int	NN
games_played	int	NN
winning_percentage	float	NN
streak	int	NN

2. Relationships

Show how tables relate to each other using connecting lines or arrows.

Examples:

- One-to-many: a user can participate in many games
- Many-to-many: players can be in multiple games (handled via a join table)

Use clear relationship notation (e.g., 1–*, –, 1–1) to show cardinality.

Lobby - User (1:M)
Each lobby can have many users. Each user can only be in one lobby at a time (join table).

User - Game (M: M)
Each user can play many games, and every game can have many users.

Statistics - User (1:1)
Each set of statistics belongs to only one user, and each user only has one set of statistics.

Lobby - Games (1:M)
Each lobby may have many games, but each game only originates from one lobby.

Game – Cards (1:M)
Each game only uses one deck filled with many cards; the used cards only belong to that particular game.

User - Game participant (1:M)
Each user can show up in many games as a participant

Game - Game participant (1:M)
Each game can have many game participants, but each game participant belongs to one game.

Game - Chat (1:1)
Each game only has one global chat, and each chat belongs to one game.

User - Chat (1:M)
A user can send multiple chats, but each chat only belongs to one user

Game - Move (1-M)
Each game can have many moves played, but each move is only played in one game

User - Move (1-M)
Each user can have many moves, but each move belongs to one user

3. Normalization and Design Rationale

In a brief note (1 short sentence), explain:

- Why you structured the tables this way
- How this design supports the features you identified in Milestone 1
- Any tradeoffs or simplifications you made for the MVP

We structured the tables around clear entities like users, lobbies, games, and moves that support classic multiplayer UNO gameplay. This design supports core Milestone 1 features like room creation, joining games, tracking moves, and recording results. In our original MVP we made an effort to keep the design simple and focused. We tried to do the same thing here with our schema.