📘 Memory Maze Game – Backend Summary

🔹 1. Backend Features Completed

- **Dynamic Level Generation**
  Levels are generated on the fly with increasing grid size, tile complexity, and decreasing timer.
- **Maze Pattern Generation**
  Each level has a unique pattern to memorize with varying grid layouts (3x3, 4x4, 5x5, etc.).
- **Obstacles and Lifelines**
  - Obstacles are randomly placed in each level.
  - Hitting an obstacle deducts one of 3 available lifelines.
  - Lifelines are tracked at each level start and update dynamically.
- **Power-Ups System**
  - Power-ups are awarded for completing 3–4 levels in a row **without losing a lifeline**.
  - Types of Power-ups:
    - 🎯 **Reveal Tile**: Shows one correct tile.
    - 🛡️ **Extra Life**: Adds an extra lifeline.
    - ⏸️ **Freeze Timer**: Adds time or pauses countdown.
    - ⏭️ **Skip Level**: Allows skipping the current level.
- **API Returns All Data Needed for Rendering**
  Each level response includes:
  - Level number
  - Grid size
  - Maze tile positions
  - Timer limit
  - Obstacles
  - Lifelines

🔹 2. Sample API Endpoint

GET /api/levels/:levelNumber

✅ Example Response (Level 3)

```
{
 "level": 3,
 "grid_size": 3,
 "sequence_length": 4,
 "time_limit": 14.1,
 "maze_positions": [[0,1],[2,2],[1,0],[2,1]],
 "obstacles": [[0,2],[1,1]],
 "lifelines": 3
}
```

✅ Example Response (Level 10)

```
{
 "level": 10,
 "grid_size": 5,
 "sequence_length": 8,
 "time_limit": 12,
 "maze_positions": [[1,0],[3,4],[0,2],...],
 "obstacles": [[1,1],[2,2],[0,3]],
 "lifelines": 3
}
```

🔹 3. Folder Structure of Backend

Folder/File	Description
controllers/	Handles core game logic like level creation, user interaction, and power-up awarding.
routes/	Defines all API endpoints and connects them to controller methods.
models/	MongoDB schema definitions using Mongoose (User, Level, Powerup).
utils/	Utility functions to generate mazes, shuffle arrays, and create timers.
config/	Configuration files including database connection.
middlewares/	Authentication, validation, and error-handling middleware.
server.js	Entry point of the app, sets up Express server and routes.

🔹 4. MongoDB Database Setup

- **Database**: MongoDB Atlas (Cloud-hosted)
- **ODM Library**: Mongoose

📁 Collections:

- users:
  Stores player info, progress, lifeline count, and power-up inventory.
- levels:
  Contains static level templates or logs for level completions if needed.
- powerups:
  Tracks when a power-up is earned, used, and the type.

⚙️ Database Config:

- DB Connection handled inside config/db.js.
- .env stores the MongoDB URI as MONGO_URI.