**Project Documentation: Console Game Management System**

**1. Introduction**

This project is a **Console Game Management System** consisting of three main components:

* **Main Console (Dashboard):** Receives and displays real-time score updates.
* **Old Console:** Runs an automatic scoring system and sends periodic updates.
* **New Console:** Allows manual score increment using the + key and sends updates to the dashboard.

**2. System Architecture**

The system follows a **client-server** model where:

* **Old and New Consoles act as clients** that send updates to the Main Console.
* **Main Console acts as a server**, listening for updates and displaying scores.
* **Communication is done using HTTP requests** with JSON payloads.

**3. Components & Code Overview**

**3.1 Main Console (Dashboard)**

The Main Console is responsible for:

* Receiving updates from the consoles.
* Fetching scores from the Old Consoles.
* Displaying real-time status of all consoles.

**Main Functionality:**

* Starts an HTTP server (http://localhost:5000/update).
* Stores and manages the list of active consoles.
* Stops inactive consoles if no updates are received for more than 10 seconds.

**3.2 Old Console**

The Old Console:

* Runs on a dynamically assigned port.
* Automatically increments the score every 3 seconds.
* Hosts a small HTTP server that provides score updates to the Main Console.
* Sends periodic updates to the Main Console every 5 seconds.

**3.3 New Console**

The New Console:

* Generates a unique console number and code at startup.
* Allows the user to enter a player name.
* Increases the score manually using the + key.
* Sends score updates to the Main Console.

**4. How to Run and Test the System**

**Step 1: Start the Main Console**

1. Open a terminal or command prompt.
2. Navigate to the Main Console project directory.
3. Run the program:

dotnet run

1. The dashboard should display: Main Board Started!

**Step 2: Start an Old Console**

1. Open another terminal window.
2. Navigate to the Old Console directory.
3. Run the program:

dotnet run

1. The console will display a randomly assigned port and start incrementing the score automatically.
2. It will send updates to the Main Console every 5 seconds.

**Step 3: Start a New Console**

1. Open another terminal window.
2. Navigate to the New Console directory.
3. Run the program:

dotnet run

1. Enter a player name when prompted.
2. Press the + key to manually increase the score.
3. Each time you press +, the score is updated and sent to the Main Console.

**Step 4: Monitor the Main Console**

* The Main Console should display:
* [New Console] Console New-1234 | Code: N567890 | Player: John | Score: 3 | Status: Running

[Old Console] Console Old-5678 | Code: O123456 | Player: Jane | Score: 10 | Status: Running

* If an Old Console stops sending updates, its status will change to Stopped.
* If a New Console stops sending updates for more than 10 seconds, its status will change to Stopped.

**5. Data Format & Communication**

**5.1 JSON Payload Sent to Main Console**

{

"Number": "New-a1b2c3d4",

"Code": "N738291",

"PlayerName": "JohnDoe",

"Score": 5,

"IsOld": false

}

**5.2 API Endpoints**

|  |  |  |
| --- | --- | --- |
| **Endpoint** | **Method** | **Description** |
| /update | POST | Receives console updates |
| /details (Old) | GET | Fetches Old Console details |

**6. Expected Output & Behavior**

|  |  |
| --- | --- |
| Action | Expected Behavior |
| Start Main Console | Displays "Main Board Started!" and waits for updates |
| Start Old Console | Generates an ID and increments score automatically |
| Start New Console | Generates an ID, allows manual score updates |
| Press + in New Console | Increases score and updates Main Console |
| Stop sending updates (New) | Status changes to "Stopped" after 10 seconds |
| Stop Old Console | Status changes to "Stopped" on Main Console |

**7. Conclusion**

This Console Game Management System provides an efficient way to track scores in real-time. By using HTTP-based communication, it ensures seamless integration between consoles and the Main Dashboard. The system can be extended further by implementing features such as leaderboard storage or player authentication.

**8. Future Improvements**

* Implement a Web UI instead of console-based output.
* Add database storage for score history.
* Improve error handling and logging.
* Support multiple Main Consoles for load balancing.
* We can use **RabbitMQ** to handle message-based communication instead of multiple ports. And to make better performance.