

|  |
| --- |
| Next Games |
| Chat Groups System Design |
| [Document subtitle] |

|  |
| --- |
| Mohamed Hamed  1-5-2020 |

Contents

[Solution Approach 2](#_Toc28970401)

[System Architecture 2](#_Toc28970402)

[Solution Architecture 3](#_Toc28970403)

[Client App 4](#_Toc28970404)

[Solution Extension and Scale-out 5](#_Toc28970405)

# Solution Approach

The solution was implemented by SignalR to handle all the communication between clients, Azure Cosmos DB to store all the data, and a loosely coupled Web APIs between SignalR Layer and Cosmos DB Layer.

# System Architecture

The system consists of the following components:

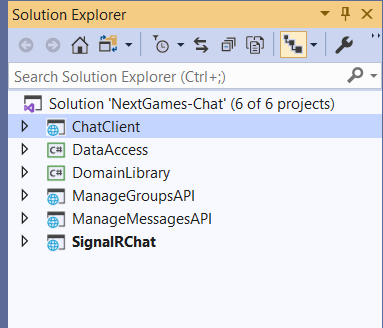
1. Azure SignalR Service which responsible for all communications between clients based on WebSocket.
2. REST WebAPIs Services acts as layer in between SignalR Service and the DB which responsible for all communications between SignalR Service and the DB for READ/WRITE/UPDATE operations and Data persistence.
3. Azure Cosmos DB which responsible for storing all data like users, groups, chat conversation history.
4. Web App client as an interface which allows clients to Create, Join, leave chat groups and chat with each other inside chat groups.



# Solution Architecture

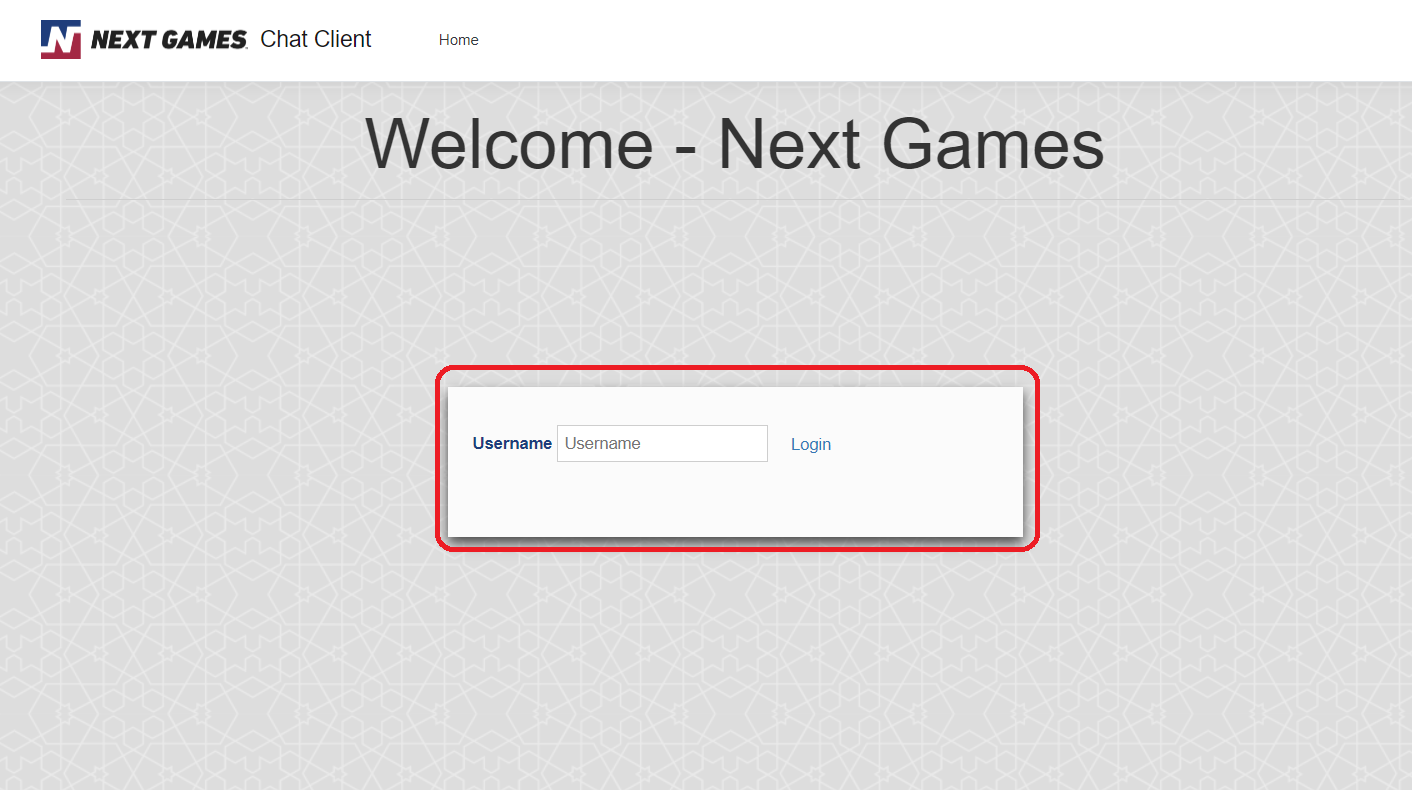
The solution consists of 6 projects as follow:

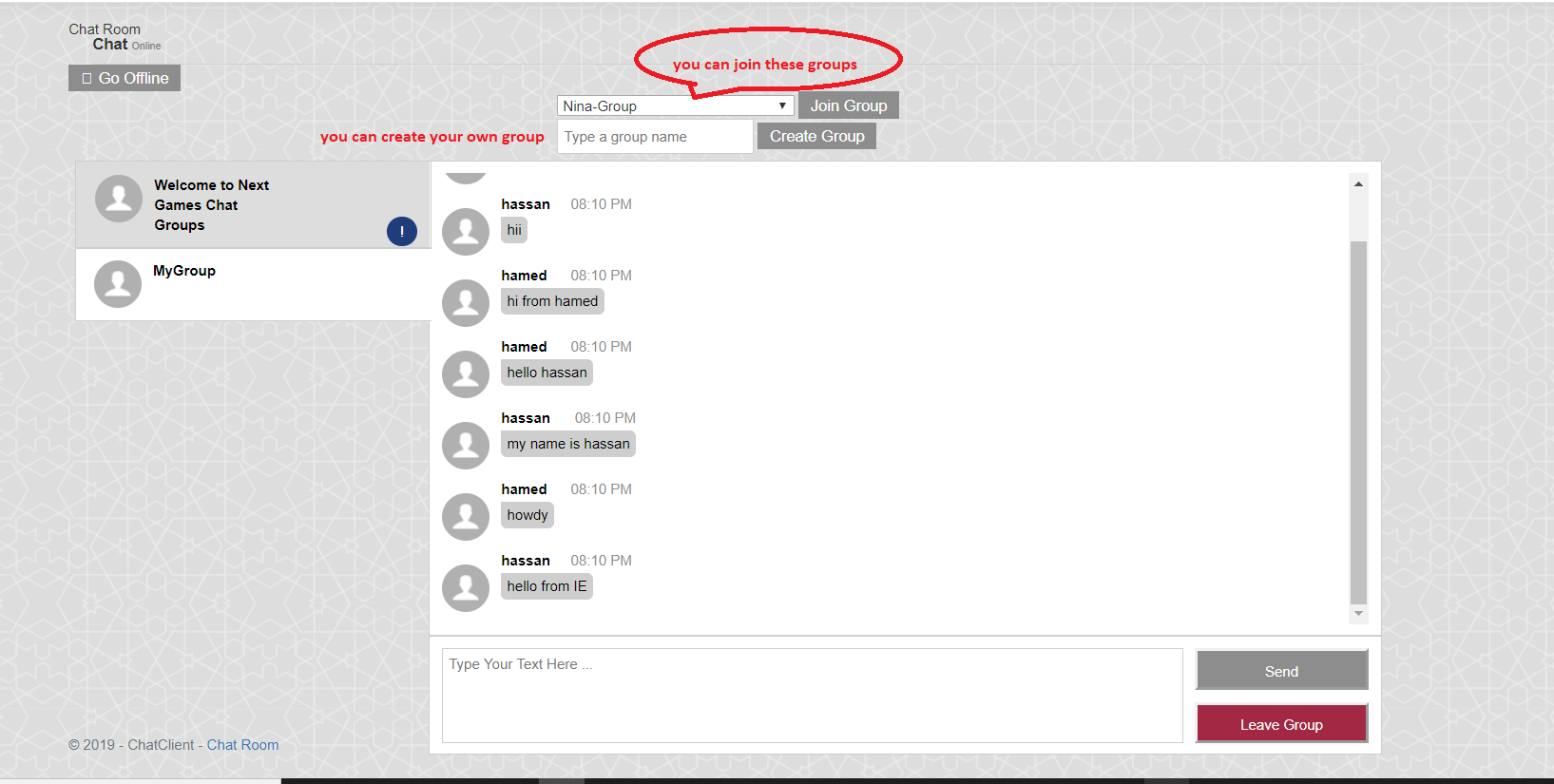
|  |  |
| --- | --- |
| SignalRChat | Responsible for all *SignalR* logic and holds the *ChatHub* engine. **Published on Azue SignalR Service**. |
| ChatClient | .Net Core 3.0 Web Application to allow clients to connect to *ChatHub* and communicate with each other. **Published on Azure App service**. |
| ManageGroupsAPI | .Net Core 3.0 Web API Project responsible for all groups and user services: join, add, leave groups. **Published on Azure App service.** |
| ManageMessagesAPI | .Net Core 3.0 Web API Project responsible for all Messages services: Save message and Retrieve Groups Messages. **Published on Azure App service** |
| DataAccessLayer | .Net Core 3.0 Class Library project responsible for accessing Azure Cosmos DB **Containers**: Users, Groups, Messages. And Implements *DomainLibrary.IRepository.* Dependency Injection is done in Web API projects startup for this implementation. |
| DomainLibrary | .Net Core 3.0 Class Library acts as a business layer which contains all business functions in IRepository |



# Client App

1. When user will run the client web app, he will be asked to enter his user name which he wants to use.
2. Then he will be redirected to the chat screen which contains all chat groups the user can join and also, he can create new chat group.
3. Then he will be able to chat with other users in each group he joins.





# Solution Extension and Scale-out

**Extension**: The solution is now ready to run on only one game, although it is designed to support multiple games as it has a game dimension on the DB level which will allow the Users, Groups, Messages to be different for each game.

**Scale-out**: The solution can be easily scaled-out and new features can be added because it designed with loosely coupled components, new features can be added easily without any affect to other components and all app services can be scaled-out to support as many users and request as they join.

