**NetChallenge Application**

NetChallenge is a real-time chat application built using ASP.NET Core SignalR and RabbitMQ. It includes a bot (StockBot) that fetches stock quotes from an external API when a user submits a command like /stock=stock\_code.

**Features**

* Real-time messaging between multiple users using SignalR.
* StockBot integration to fetch stock quotes from the [Stooq API](https://stooq.com/).
* Messages include timestamps.
* Background service (StockBotService) listens for stock quote requests via RabbitMQ.

**Prerequisites**

**1. Required Software**

* [.NET 9 SDK](https://dotnet.microsoft.com/download/dotnet/9.0)
* [Erlang/OTP](https://www.erlang.org/downloads)
* [RabbitMQ Server](https://www.rabbitmq.com/download.html)
* A web browser to access the chat interface.

**2. Install RabbitMQ**

* Install Erland/OTP
* Install RabbitMQ and ensure the management plugin is enabled:
* rabbitmq-plugins enable rabbitmq\_management
* Start RabbitMQ:
* rabbitmq-server
* Access the RabbitMQ Management Console at [http://localhost:15672](http://localhost:15672/).  
  Default credentials:
  + Username: guest
  + Password: guest.

**Getting Started**

**1. Clone the Repository**

Clone this repository to your local machine:

git clone https://github.com/your-repo/NetChallengeJobsity

cd NetChallenge

**2. Build the Application**

Restore the NuGet packages and build the project:

dotnet restore

dotnet build

**3. Run the Application**

Start the application:

dotnet run

**4. Access the Application**

Open your browser and navigate to https://localhost:7157

**How It Works**

**Chat Functionality**

**1. Real-Time Messaging**

* **Users can join the chatroom and send real-time messages to other users.**
* **Messages are displayed with timestamps.**

**2. Multiple Chatrooms**

* **Users can create or join different chatrooms.**
* **Each chatroom functions independently, with its own set of messages and participants.**
* **To switch between chatrooms, users simply select or create a chatroom via the interface.**

**3. StockBot Command**

* **In any chatroom, users can type /stock=stock\_code (e.g., /stock=aapl.us) to fetch stock quotes. This command can be use in any part of the message.**
* **The bot processes the request and replies with the stock's latest closing price e.g.,**
  + **AAPL.US quote is $246.75 per share.**

**Message Workflow**

1. **A user sends a message or command in a specific chatroom.**
2. **The message is broadcasted to all participants in that chatroom via SignalR.**
3. **For /stock=stock\_code commands:**
   * **The command is sent to RabbitMQ via the RabbitMqMessageQueue.**
   * **The StockBotService processes the request and sends the response to the appropriate chatroom.**

**Technical Details**

**Architecture**

* **Backend**:
  + ASP.NET Core with SignalR for real-time messaging.
  + RabbitMQ for decoupled messaging between the bot and chat.
* **Frontend**:
  + Razor views with JavaScript for SignalR client integration.

**Key Classes**

* **ChatHub**: Handles real-time chat functionality using SignalR.
* **StockBotService**: A background service that listens for stock requests and responds with quotes.
* **RabbitMqMessageQueue**: Manages communication with RabbitMQ.

**Testing**

**Unit Tests**

* The application includes unit tests for critical functionalities:
  + ChatHub: Ensures messages are broadcasted with timestamps.
  + StockBotService: Tests stock quote parsing logic.

**Troubleshooting**

1. **RabbitMQ Connection Issues**:
   * Ensure RabbitMQ is running on localhost and the stock\_requests queue is available.
   * Verify RabbitMQ logs for connection errors.
2. **API Issues**:
   * If the Stooq API is unreachable, ensure you have internet access and verify the API URL.
3. **SignalR Issues**:
   * If real-time messages aren’t working, check the browser console for errors and ensure SignalR scripts are correctly included.