Create digital version of Resource management game (RMG-1)

[RMG-11] Communication Channel Created: 2024-09-26 Updated: 2024-09-30

Status:	To Do
Project:	Resource Management Game
Components:	None
Affects versions:	None
Fix versions:	None
Parent:	Create digital version of Resource management game

Type:	Sub-task	**Priority:**	Major
Reporter:	Shilpa Ramisetti	**Assignee:**	Unassigned
Resolution:	Unresolved	**Votes:**	0
Labels:	None		
Remaining Estimate:	Not Specified		
Time Spent:	Not Specified		
Original estimate:	Not Specified		

Description

Display a communication window at the bottom-right of every game page

Backend

1. **Implement WebSocket Configuration:**
   - Set up WebSocket support in the Spring Boot application.
   - Create a web socket topic.
   - Create a WebSocket configuration class to register the `/chat` endpoint, which will handle the real-time communication between players.
2. **Implement WebSocket Handler:**
   - Develop a `ChatWebSocketHandler` class that manages WebSocket connections.
   - Handle incoming messages from players, broadcasting them to all connected clients.
   - Ensure that when a player disconnects, their session is properly closed and removed from the active session list.
3. **Extend Game Data Model:**
   - Integrate chat and game-related messaging into the existing in-memory data model.
   - Track chat messages alongside player states and game actions to ensure consistent and synchronized communication.
4. **Implement Real-time Game State Updates:**
   - Use WebSocket messages to broadcast updates about player actions, role assignments, and other game events in real-time to all connected clients.
   - Ensure the WebSocket server can handle high concurrency, with multiple players sending and receiving messages simultaneously.
5. **Implement Security and Validation: (nice to have)**
   - Validate all incoming WebSocket messages to ensure they are from authenticated players within the game session.
   - Implement appropriate security measures to prevent unauthorized access to the communication channel.

Frontend

1. **Establish WebSocket Connection (On Game Create/Join): [SVE]**
   1. Subscribe to web socket topic for respective players.
   2. Set up a WebSocket connection to the `/chat` endpoint when a player joins the game.
   3. Ensure that the WebSocket connection is managed correctly, handling connection open, message reception, and connection close events.
2. **Display Real-time Chat Interface:**
   - Develop a chat component that will be integrated into the existing game UI, allowing players to send and receive messages in real-time.
   - Ensure that the chat interface is responsive and does not interfere with other game UI elements.
3. **Display Real-time Messages:**
   - Render incoming chat messages in the chat component, updating the view as new messages arrive.
   - Maintain the order of messages and ensure that the chat is scrollable, with the latest messages visible at the bottom.
4. **Implement Chat in Moderator and Player Views:**
   - **Moderator View:** Enable the Moderator to participate in the chat, with the ability to send messages and view all player communications.
   - **Player View:** Enable players to communicate with each other and the Moderator in real-time through the chat, in addition to viewing the game state updates.
5. **Integrate Real-time Game Notifications:**

- Use WebSocket messages to display real-time game notifications within the chat or as separate UI elements (e.g., "Player 1 has joined," "Project 2 assigned to Project Manager 3").
- Ensure that these notifications are visible to both the Moderator and players, keeping everyone informed of the game's progress.

Reference Link: [SVE]

https://spring.io/guides/gs/messaging-stomp-websocket