System Design Document

1. Class Name: GamePage.tsx

- Parent Class: N/A
- Subclasses: N/A
- Responsibilities:
    - Render main game interface
    - Handling the user input
    - Displaying game prompts and the game state
- Collaborators:
    - button.tsx
    - card.tsx
    - input.tsx

2. Class Name: userList.tsx

- Parent Class: N/A
- Subclasses: N/A
- Responsibilities:
    - Displays the list of users in the game
- Collaborators:
    - N/A

3. Class Name: SignIn.tsx

- Parent Class: N/A
- Subclasses: N/A
- Responsibilities:
    - Displays a user interface for signing in
    - Integrates the Clerk api for authentication
- Collaborators:
    - button.tsx (components/ui/button.tsx). Used for the sign-in process

4. Class Name: Header.tsx

- Parent Class: N/A
- Subclasses: N/A
- Responsibilities:
    - App header which displays the title, and user authentication status
- Collaborators:
    - mode-toggle.tsx. Which allows the user to switch display themes

5. Class Name: mode-toggle.tsx

- Parent Class: N/A
- Subclasses: N/A
- Responsibilities:
    - Allows the user to switch between light mode, dark mode, or the system theme
- Collaborators:
    - button.tsx : Used for the theme option button

6. Class Name: SignedIn.tsx

- Parent Class: N/A
- Subclasses: N/A
- Responsibilities:
    - Displays a welcome message and options for the user once signed in
- Collaborators:
    - button.tsx: used for the navigation buttons

7. Class Name: Room.tsx

- Parent Class: N/A
- Subclasses: N/A
- Responsibilities:
    - Display a list of active rooms.
    - Allow users to join or leave a room.
- Collaborators:
    - roomModel.js for backend data management.
    - userList.tsx for displaying participants.

8. Class Name: MessageBoard.tsx

- Parent Class: N/A
- Subclasses: N/A
- Responsibilities:
    - Display real-time messages in a game room.
    - Allow users to send new messages.
- Collaborators:
    - roomMsgModel.js for managing message data.

Description of System Interaction with the Environment

Dependencies and Assumptions:

- Operating System: Cross-platform (Windows, macOS, Linux)
- Programming Language: Typescript (React for frontend, Node.js for backend)

- Database: MongoDB (using the cloud) for storing user, room, and message data.
- Network Configuration: Assume good internet connection for the API calls and database access.
- Authentication: Using Clerk for user authentication and session management for the users

Diagram: