Software Requirements Specification: Clue-Less

1. Introduction

1.1 Purpose

This document outlines the detailed requirements for the Clue-Less system, encompassing both functional and non-functional criteria, along with constraints and definitions of key terms.

1.2 Scope

This specification applies to the entire Clue-Less system.

2. Glossary

- **Player**: A registered user who participates in the game.
- **Subsystem**: A secondary system within the main system, responsible for specific functionalities.
- **Interface**: A point where two systems, subjects, organizations, etc., meet and interact.
- **Virtual Machine (VM) - Azure Cloud:** A virtualized computing instance provided by Microsoft Azure, which allows users to run applications and host operating systems in a cloud-based environment. Azure virtual machines provide flexibility, scalability, and on-demand availability for various computing needs.
- **Server-to-Client Communication Waiting Time:** The maximum duration it takes for data to travel from the server to the client during interactions, ensuring responsive user experiences.
- **Usability:** The measure of how user-friendly and efficient the software is for end-users, encompassing aspects like user interface design and accessibility.
- **Scalability:** The ability of the software system to handle increased user loads and growing demands by adding resources or nodes, maintaining performance and responsiveness.
- **Uptime:** The duration during which a system or service is operational and available for use, often expressed as a percentage of total time to indicate reliability and availability.

3. Software Architecture

3.1 Subsystem Identification

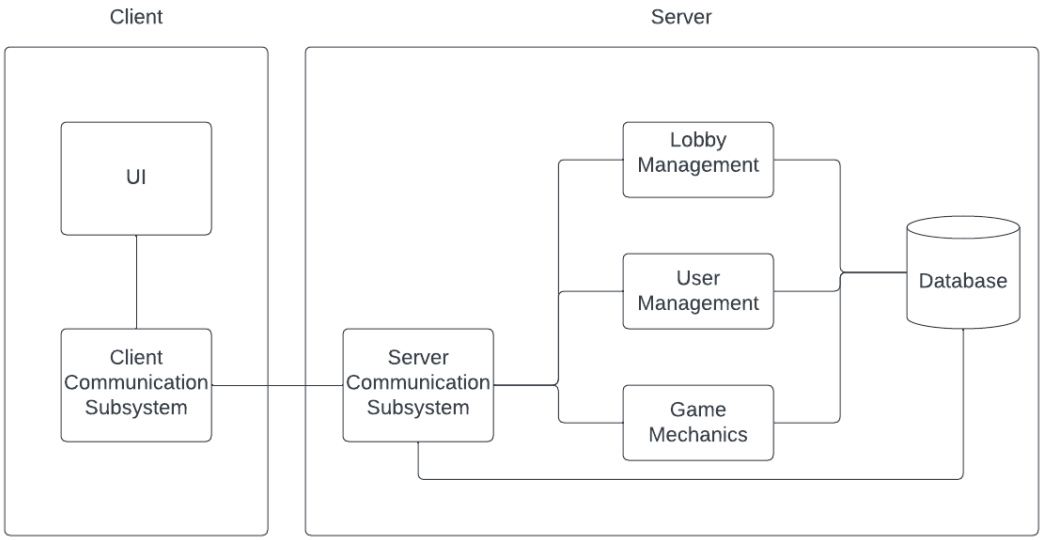
Client:

- **User Interface (UI) Subsystem**
- **Communication Subsystem (Client-Side)**

Server:

- **Communication Subsystem (Server-Side)**
- **User Management Subsystem**
- **Game Mechanics Subsystem**
- **Lobby Management Subsystem**
- **Database Subsystem**

3.2 Subsystem Relationship



Client Side:

- **User Interface (UI) Subsystem**
  - Interacts With: Communication Subsystem
  - Actions:

        - **Sends:** User actions, game moves, chat messages
        - **Receives:** Game state updates, chat messages
- **Communication Subsystem**
    - Interacts With: UI Subsystem, Server's Communication Subsystem
    - Actions:
        - **Sends:** User actions, game moves, chat messages to Server
        - **Receives:** Game state updates, chat messages from Server

Server Side:

- **Communication Subsystem (Server)**
    - Interacts With: All other server subsystems, Client's Communication Subsystem
    - Actions:
        - **Sends:** Game state updates, chat messages to Client
        - **Receives:** User actions, game moves, chat messages from Client
- **User Management Subsystem**
    - Interacts With: Communication Subsystem, Database Subsystem
    - Actions:
        - **Sends:** User login status, profile data
        - **Receives:** User registration details, login requests
- **Lobby Management Subsystem**
    - Interacts With: Communication Subsystem, Database Subsystem
    - Actions:
        - **Sends:** Lobby creation success, current lobby state
        - **Receives:** Lobby join requests, lobby creation requests
- **Game Mechanics Subsystem**
    - Interacts With: Communication Subsystem, Database Subsystem
    - Actions:
        - **Sends:** Game state updates, game results
        - **Receives:** Game moves, game start requests
- **Database Subsystem**
    - Interacts With: User Management Subsystem, Game Mechanics Subsystem, Lobby Management Subsystem
    - Actions:
        - **Sends:** Stored user data, saved game states, lobby data
        - **Receives:** Requests to save/update user data, game states, lobby data

3.3 Information Domain, Functional Domain, Interfaces

Subsystem	Information Domain	Functional Domain	Interfaces
User Interface (UI) Subsystem	Contains visual elements and controls allowing players to interact with the game.	Display game state, capture player actions, show chat messages, show errors or prompts.	User input events (click, type, etc.), display updates, error prompts.
Communication Subsystem (Client-Side)	Handles client-side protocol details, messages being sent to the server, and updates received.	Send player actions to server, receive real-time updates, update UI in real-time.	send_message(), fetch_game_state(), send_player_move(). (JSON)
Communication Subsystem (Server-Side)	Handles server-side protocol details, processing client messages, and sending real-time updates.	Listen for client messages, process incoming requests, send updates to connected clients, manage server-side WebSocket connections.	WebSocket events like connection, disconnection, receive_message(), send_update.() (JSON)
User Management Subsystem	Contains user profiles, status data, current game sessions, and user history.	Register new users, authenticate existing users, manage user sessions, maintain user game status.	register_user(), login_user(), logout_user(), get_user_status().
Game Mechanics Subsystem	Contains game rules, active game states, player moves, and win/loss conditions.	Process player moves, evaluate win/loss conditions, progress game to next state, manage turn-based logic.	process_move(), evaluate_game_state(), start_new_game().
Database Subsystem	Contains data models, schemas, saved game states, user profiles, and historical game data.	Save game states, load historical game data, save user profiles, manage data integrity.	save_game(), load_game(), save_user_profile(), fetch_user_profile().
Lobby Management Subsystem	Contains data models for lobbies, player counts, lobby status	Create new lobbies, manage players in lobbies, handle player matchmaking,	create_lobby(), join_lobby(), leave_lobby(), get_lobby_status().

Subsystem	Information Domain	Functional Domain	Interfaces
	(waiting, in-progress), and matchmaking logic.	manage lobby lifecycles.	

4. Functional Requirements

4.1 Use Cases

4.1.1 Sequential Gameplay Use Cases

Summary Information	
Use Case Name	User Register and Login
Use Case Number	UC_001
Use Case Goal	User registers (if not registered, see UC_008) and logs into the system
Trigger	User inputs information and registers/logs in
Actors	User, database
Pre-conditions	
Post-conditions	User will be logged in and open lobbies will be displayed
Main Success Scenario	
Actor Action	System Response
1. User enters user information	2. System validates user information
	3. System logs user in
Alternate Scenarios	
System finds user login information is invalid	
	1. System displays login error 2. System prompts user to try to login once more.

Summary Information	
Use Case Name	Lobby Creation and Player Joining
Use Case Number	UC_002

Use Case Goal	User creates game lobby, other users join, lobby owner starts game when desired number of players join
Trigger	Creating/joining a lobby
Actors	User, lobby management system
Pre-conditions	User must be logged in
Post-conditions	User will join a lobby and others in the lobby will be displayed
Related Use Cases	UC_001
Main Success Scenario	
Actor Action	System Response
1. User starts lobby creation	2. System gives lobby creation options
3. User chooses options and creates lobby	4. System displays lobby information
Steps 5-6 repeated until lobby owner starts game	
5. User joins lobby	6. System adds user to lobby information display
Alternate Scenarios	
System detects that lobby is full (at step 5):	
	1. System displays lobby is full message
User leaves lobby (after step 6)	
1. User leaves lobby	2. System removes user from lobby information display

Summary Information	
Use Case Name	Character Selection
Use Case Number	UC_003
Use Case Goal	User chooses character from available options
Trigger	Creating/joining a lobby
Actors	User, Character availability system
Pre-conditions	User must be in a lobby
Post-conditions	User will choose a character and it will be displayed to all players in the lobby
Related Use Cases	UC_001, UC_002
Main Success Scenario	
Actor Action	System Response
Steps 1-2 repeated until game starts	
1. User in lobby selects character	2. System displays the available characters

Summary Information	
Use Case Name	Game Initialization
Use Case Number	UC_004
Use Case Goal	System distributes cards to each player, game board is presented to players
Trigger	Game start by lobby owner
Actors	Game Mechanics system
Pre-conditions	There must be at least two players in a lobby to start game Players have all selected a character Lobby owner has started the game
Post-conditions	Game board will be displayed to all players in the lobby
Related Use Cases	UC_002, UC_003
Main Success Scenario	
Actor Action	System Response
	1. System shuffles cards and divides them up to the users
	2. System displays users cards to them
	3. System displays game board to users

Summary Information	
Use Case Name	Turn-based Gameplay
Use Case Number	UC_005
Use Case Goal	Players take turns in order, of which they can move, make a suggestion, or make an accusation
Trigger	Game start or previous player finishes turn
Actors	User, rule enforcement system
Pre-conditions	The game is not over
Post-conditions	Users turn will be displayed to other players and next user will have their turn
Related Use Cases	UC_004
Main Success Scenario	
Actor Action	System Response

Steps 1-5 repeated until game ends	
	1. System prompts user that it is their turn
2. User takes turn moves makes a suggestion makes an accusation	3. System displays move/suggestion/accusation to all players
	4. If the user made a suggestion or accusation, the system resolves/facilitates a resolution of the suggestion/accusation
	5. System updates the next player in turn order

Summary Information	
Use Case Name	Filling out Note Sheet
Use Case Number	UC_006
Use Case Goal	User selects to fill out x's on the note sheet or remove them.
Trigger	User selects a notes icon to bring up the note sheet interface.
Actors	User, Note Sheet UI
Pre-conditions	It is the character's turn, the notes sheet UI is not currently open.
Post-conditions	The note sheet will be minimized and save the changes that the player has made
Main Success Scenario	
Actor Action	System Response
1.User selects the note sheet menu	2. System display note sheet in UI with player's current notes
3. Player chooses to mark x's or unmark x's	4. System logs information and reflects the changes visually
5. Player chooses to close window	6. The window is closed, saving the player's current sheet for future use.

Summary Information	
Use Case Name	Game is Ended
Use Case Number	UC_007

Use Case Goal	The game displays the winner and prompts to begin another game.
Trigger	A player's accusation is correct and the game ends.
Actors	User, UI, Server
Pre-conditions	The game has ended.
Post-conditions	A new game will begin or not, and the previous game's data will be saved.
Main Success Scenario	
Actor Action	System Response
	1.The game ends causing the system to display the winner along with the correct accusation.
	2. After allowing the player time to process, the system then prompts each player to play another game or quit.
3.Each player chooses to play another game or to quit	4.Enough players choose to play a game and so the system begins the process of starting a new game.
	5.The previous game's data is saved to the database
Alternate Scenarios	
Not enough player decided to play again	
	1.The game ends causing the system to display the winner along with the correct accusation.
	2. After allowing the player time to process, the system then prompts each player to play another game or quit.
3.Each player chooses to play another game or to quit	4.Not enough players choose to play again and so the game informs each player that another game will not be played.
	5.The system returns players to the creating/joining menu and saves the previous game's data to the database.

4.1.2 Non-Gameplay Related Use Cases:

Summary Information

Use Case Name	Creating a new Account
Use Case Number	UC_008
Use Case Goal	If the user does not have an account they create a new one.
Trigger	The user chooses to create an account.
Actors	User, Database
Pre-conditions	The user does not have an account.
Post-conditions	The user's account information will be saved.
Main Success Scenario	
Actor Action	System Response
1. User chooses to create new account	2.The system display requirements for username and password
3.The user fills out the desired username and password.	4.The system validates and saves this information in the database for future use
	5.The user is informed of successful account creation and enters the main menu.
Alternate Scenarios	
The user's desired username is taken.	
1.User chooses to create new account	2.The system display requirements for username and password
3.The user fills out the desired username and password.	4.The system attempts to validate information, but the username is already in use.
	5.The user is prompted to change the desired username.
6.User enters a new username.	7.The system validates and saves this information in the database for future use.

Summary Information	
Use Case Name	Accessing Game History
Use Case Number	UC_009
Use Case Goal	Display current user's game history.
Trigger	The user opens the game history menu.
Actors	User, Database
Pre-conditions	The user has selected to open the game history from the main menu.

Post-conditions	The user's game history is displayed.
Main Success Scenario	
Actor Action	System Response
1.The user opens the game history menu from the main menu	2. The system accesses the database to query the games saved under the current user's username.
	3. This query creates a list of the users 10 previous games along with game information such as the winner, players, and final accusation.
	4. The system displays this information in the UI.

4.2 Supplementary Functional Requirements

Supplementary functional requirements are the additional functions not covered by use cases that ensure the software system offers a smooth and seamless experience for users across different situations.

1. **Session Restoration**:
    - In case a player accidentally closes the game or faces an unexpected interruption, the system should allow them to restore and resume their gameplay from the point they left off. This ensures that players don't lose their progress and can continue the game without frustration.
2. **Adaptive User Interface**:
    - The game should be able to adapt its user interface based on the device it's accessed from. Whether a player is on a desktop, tablet, or mobile device, the interface should rearrange itself for the best user experience.
3. **Error Handling and User Feedback**:
    - The system should handle unexpected errors gracefully, without causing crashes or major disruptions to the user. Any errors encountered should be logged for further investigation. Additionally, users should be provided with a mechanism to report bugs or give feedback, ensuring continuous improvement of the system.

5. Non-functional Requirements

5.1 Performance Requirements

Set the server-to-client communication waiting time to a maximum of 3 seconds to ensure responsive user interactions.

5.2 Usability

Design intuitive user interfaces and accessible features to enhance the overall user experience.

5.3 Security and Privacy

Implement password security measures, including the use of strong password policies with a minimum of 8 characters, a combination of uppercase and lowercase letters, numbers, and special characters, to enhance user account security and prevent unauthorized access.

5.4 Scalability

- Ensure the software can handle at least 6 concurrent users.
- The server virtual machine (Azure VM) should support horizontal scaling by adding new servers to the cluster seamlessly as user demand grows.

5.5 Availability

Ensure a cloud virtual machine hosting the software is available at all times, maintaining an uptime of 99.9% per month, with a maximum downtime of 2 hours for scheduled maintenance.

5.6 Backup and Recovery

Conduct daily data backups with a retention period of at least 30 days.

6. Implementation Constraints

- The game should be web-based and accessible via modern browsers.
- Continuous internet connection is required for gameplay.
- The system should not infringe upon any intellectual property rights of the original Clue board game.