Gebze Technical University Faculty of Engineering

Group 7

PiPLayGame Box

Project Final Report
https://https://github.com/PiPlayGameBox

12 January 2024

1 Group Members:

According to our requirements we created 4 teams, each member will be in 2 teams.

Group Members	**Numbers**	**Modules**
Abdullah Kırıcıoğlu	1901042604	Mobile Application, Portable Server
Abdulsamet Aslan	200104004098	Desktop Application, Portable Server
Ahmet Yazıcı	1801042639	Virtualization, Mobile Application
Elifnur Kabalcı	1801042617	Virtualization, Desktop Application
Hüseyin Sarsılmaz	1801042665	Desktop Application, Portable Server
Ozan Doruk Yavuz	200104004077	Virtualization, Mobile Application
Tuba Toprak	161044116	Virtualization, Portable Server

Table 1: Group Members and Modules

2 Project Definition:

The primary objective of this project is to engineer a portable and engaging gaming experience, accessible to users across the iOS and Android mobile platforms. Our solution combines the utilization of a Raspberry Pi device as a portable game server, a mobile application to serve as the gaming interface, and a desktop application capable of rendering gameplay in a three-dimensional (3D) environment.

Our system provides a multiplayer gaming experience in a secure and controlled manner. This security is established through the implementation of an authentication mechanism, requiring users to provide a secure password for access. A game manager, utilizing the desktop application, assumes the pivotal role of orchestrating the gaming sessions. This includes the selection of games, team assignment and the ability to manipulate game dynamics for the benefit of specific players.

Communication within our system is achieved through the use of a Wireless Local Area Network (WLAN). The Raspberry Pi device acts as a hub, creating a secure WI-FI hotspot to which all participating devices must connect. Users are required to log in with unique usernames and passwords to ensure the security of the gaming environment. Once a sufficient number of players are connected, the game manager can configure and initiate game sessions. The desktop application serves as a virtual game board, offering the game manager tools to monitor and control the game, including the ability to view the cards or hands of each player.

This project represents a fusion of hardware and software components, offering a fun gaming experience. Through the integration of a Raspberry Pi-based

game server, a mobile gaming application, and a desktop interface, we will deliver the best gaming experience we can offer.

3 Users of the System:

PiPlayGame Box consist 2 type of users.

3.1 Admin:

The Admin is a privileged user with comprehensive control over the system. This user primarily interacts with the Desktop Application to manage game-related activities.

•Login: The Admin can securely log in through the Desktop Application, ensuring access to administrative features.

•Lobby Management: Create, view, and manage game lobbies. Admin has the authority to open, close, or modify the parameters of the lobbies.

•Player Requests: Accept or decline player requests to join lobbies, ensuring control over the gaming environment.

•Statistics: Access and maintain detailed statistics for each game, tracking player performance, and other relevant metrics.

•3D Game Previews: Utilize 3D game previews to visually inspect ongoing game sessions, providing a dynamic perspective.

3.2 Players:

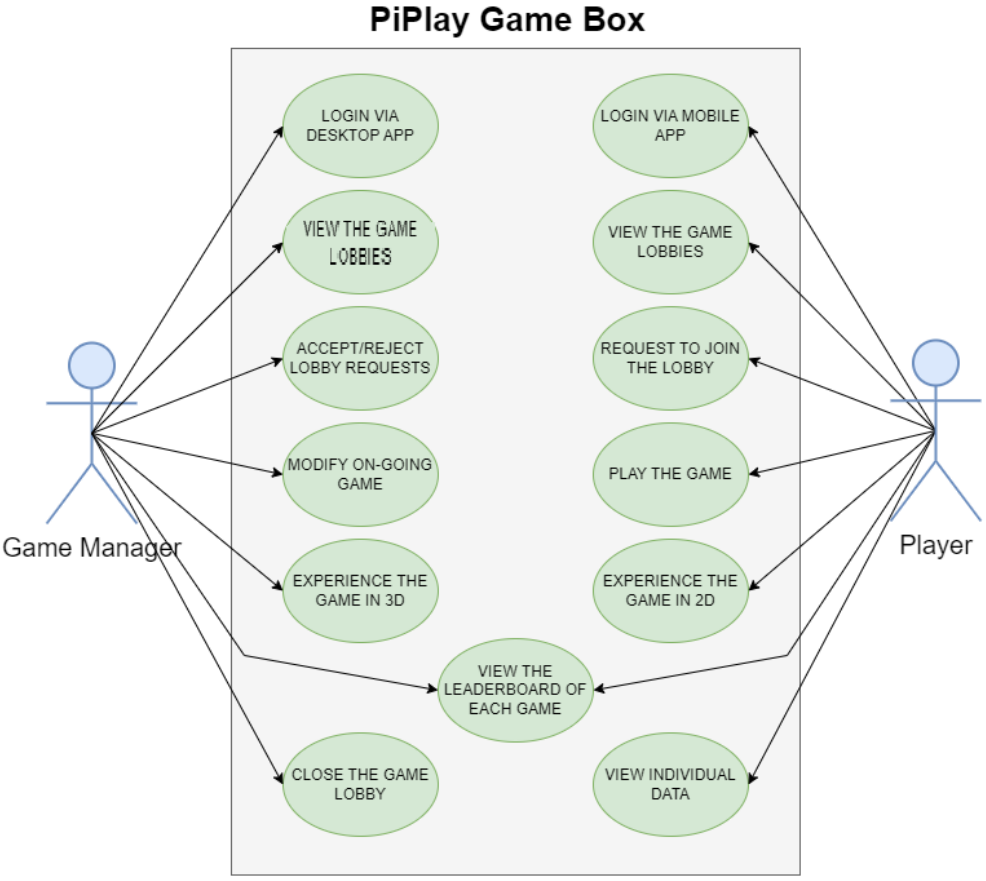
The Player is a standard user who interacts with the system primarily through the Mobile Application, engaging in gaming activities.
•Login: Players can securely log in through the Mobile Application, gaining access to personalized features.

•Lobby Interaction: View available lobbies, send requests to join lobbies, and participate in multiplayer gaming sessions.

•Gameplay: Play games in a 2D format, enjoying a seamless and immersive gaming experience on the Mobile Application.

Interaction with Admin: Send lobby requests and interact with lobby-related features facilitated by the Admin.

•Game Exploration: Explore and choose gaming options, including the ability to select different lobby types and game variations.



4 Game Descriptions:

4.1 Ludo King Game:

Ludo King is a digital adaptation of the classic board game Ludo, known for its entertaining and competitive nature. The game combines traditional gameplay with modern features for an engaging multiplayer experience.

Objective:

- The main objective is to move all your colored pawns from the starting point to the central area, known as the "home."

Gameplay:

- Players take turns rolling a die to determine the number of spaces their pawn can move.
- Pawns move clockwise around the board based on the rolled number.
- Capturing opponents' pawns is possible by landing on the same space.

Special Spaces:

- The board features special spaces such as safe zones and shortcuts, adding strategic depth to the game.

Winning:

- The first player to successfully move all their pawns to the home wins.
- To reach the home, players must roll the exact number required; otherwise, they wait for the next turn.

4.2 Okey Game:

Okey, Rectangular stones is a game of chance, logic and memory played with cues to arrange the stones. The word Okey is also the name of the joker tile in the game.

Objective:

- The goal is to complete sets of tiles, known as "melds," using specific combinations.

Game Setup:

- The game employs a set of 106 tiles, numbered from 1 to 13 in four different colors, alongside two special tiles called "jokers."
- Pawns move clockwise around the board based on the rolled number.
- Capturing opponents' pawns is possible by landing on the same space.

GamePlay:

- Players draw tiles from a central pool, aiming to form valid melds or runs of consecutive numbers in the same color.
- Jokers act as versatile tiles, capable of replacing any tile in a meld.
- Strategic discarding and drawing are crucial to completing sets and gaining an advantage.

Winning:

- The first player to successfully form all their melds declares "Okey" and wins the round.
- Points are scored based on the remaining tiles in opponents' hands.

5 Analysis and Design:

5.1 System Architecture:

The PiPlayGameBox project is built upon a modular architecture, consisting of four main modules: server, desktop application, virtualization, and mobile application. Each module plays a specific role within the system and interacts with one another.

5.1.1 Server Module:

Provides multi-user support and interacts with the database. Incorporates security and session management functionality. Manages game requests and stores game data.

5.1.2 Desktop Application Module:

Handles user login, lobby management, and game selection. Communicates with the server to retrieve live game data. Delivers a 3D gaming experience to users.

5.1.3 Virtualization Module:

Provides 3D modeling and animation for the game board and pawns. Communicates with the server to receive real-time game movements. Renders the 3D visualization of the game on the desktop application.

5.1.4 Mobile Application Module:

Manages user login, lobby viewing, and game request functionalities. Communicates with the server to fetch live game data. Offers a 2D gaming experience to users.

5.2 Database Design:

"Users" table for storing user information and sessions.
"Lobbies" table to hold lobby information for games.
"Games" table for storing game-related data.
"Leaderboard" table for tracking user scores and statistics.

5.3 User Interface Design:

5.3.1 Desktop Application:

User-friendly menus for Ludo and Okey games.
Interfaces for lobby selection and creation.
3D game board and user movements display.

5.3.2 Mobile Application:

Simple and effective user login and registration screens.
Interface for viewing lobbies and sending game requests.
2D game board and user movements display.

5.4 Security and Authorization:

Secure storage of user passwords using SHA-256 encryption.
Unique token usage for session management.
Authorized procedures for database interactions.

5.5 Data Communication and Protocols:

Secure data communication over TCP/IP.
Client-server communication using String-based protocols.

5.6 Technology and Tools:

Utilization of C++ and the Qt Framework.
QML for 3D modeling.
Process management via Git and GitHub.

6 Development and Implementation:

6.1 Server Module Development:

The server module serves as the backbone of the PiPlayGameBox project, managing user interactions, game requests, and database operations. The development process can be outlined as follows:

•Multithreaded Structure: Implemented a multithreaded structure to handle concurrent client requests efficiently.

•Database Operations: Created a database to store user information, including registration and login functionality. Implemented secure password storage using SHA-256 encryption.

•Security Measures: Integrated security features, including session tokens for user authentication.Ensured secure communication with the client by enforcing token-based authorization.

•Lobby Management: Developed functionality for creating, joining, and leaving lobbies.Enabled lobby information retrieval and display on the desktop and mobile applications.

•Real-time Game Requests: Implemented the foundation for real-time game requests and interactions.
Incorporated features for handling game-related data on the server side.

6.2 Desktop Application Module Development:

The desktop application module focuses on delivering an immersive gaming experience for users. Development highlights include.

•UI/UX Design: Created a visually appealing UI with distinct screens for login, game selection, and lobby management. Designed interactive buttons and navigation for seamless user experience.

•Virtualization Integration: Developed a virtualization module to connect the desktop application with the 3D game environment. Established communication with the server to fetch real-time game data.

•Admin Control Panel: Implemented an admin control panel with features like lobby creation, closure, and user management. Enabled the admin to view 3D game boards and user movements.

•Leaderboard Functionality: Designed a leaderboard section to showcase user scores and statistics. Incorporated functionality to retrieve and display this data from the server.

Figure 1: Login Page

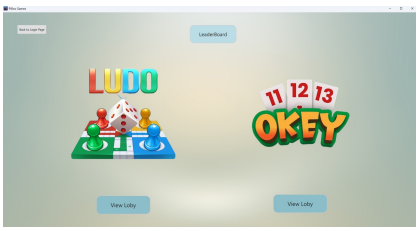


Figure 2: Lobby Screen

Figure 3: Lobby of Ludo King Game

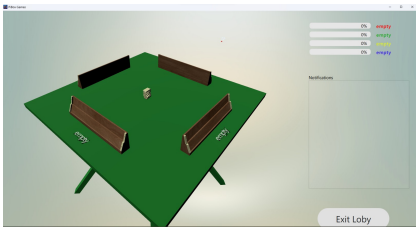


Figure 4: Lobby of Okey Game

6.3 Virtualization Module Development:

The virtualization module is crucial for rendering the 3D game environment and managing game-specific components. Development steps include:

•3D Component Design: Created 3D models for the game board, pawns, and other elements.
Designed an immersive environment for both Ludo and Okey games.

•Dynamic Camera Movement: Implemented a dynamic camera system that allows users to view the game from multiple perspectives.

•Real-time Game Data Handling: Established communication with the server to receive real-time game data.
Ensured synchronized movement of pawns based on the data received.

•Animation Effects: Developed animation effects for pawn movements and other in-game actions.
Enhanced the overall gaming experience with visual effects.

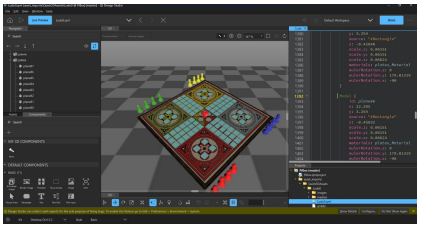


Figure 5: Qml for Ludo King Game

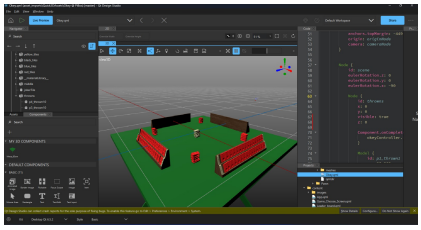


Figure 6: Qml for Okey Game

6.4 Mobile Application Module Development:

The mobile application module is designed to provide a seamless gaming experience for users on the go. Key development aspects include:

•User Authentication: Implemented user login and registration screens with secure authentication measures.
Ensured the protection of user credentials during login.

•Game Interface Design: Created an intuitive game interface with easily navigable screens.
Designed 2D game boards for Ludo and Okey, accommodating mobile screen sizes.

•Real-time Game Data Retrieval: Established communication with the server for fetching real-time game data.
Enabled players to make moves and interact with the game seamlessly.

•Animation Features: Developed animations for dice rolls and other in-game actions.

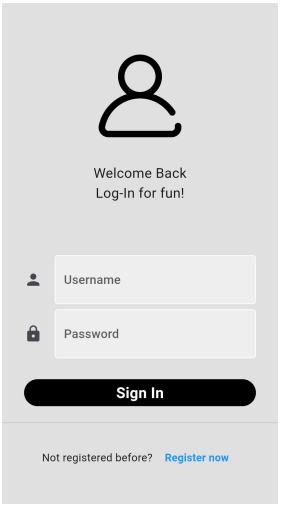


Figure 7: Login Page on Mobile

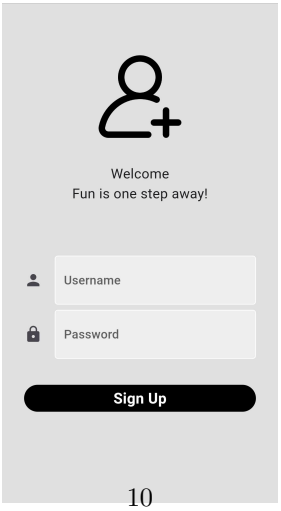


Figure 8: Sign-up Screen on Mobile

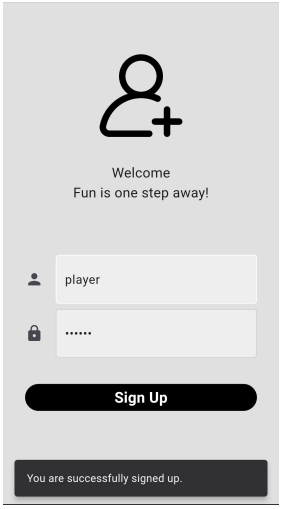


Figure 9: Sign-up Screen on Mobile

Figure 10: User profile pictures

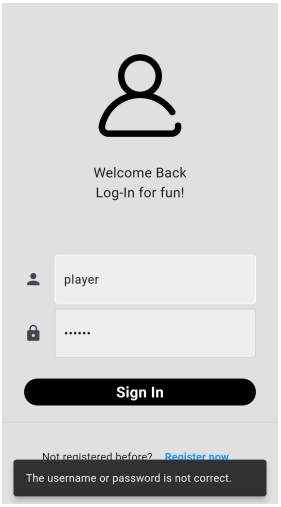


Figure 11: Sign-up Screen on Mobile

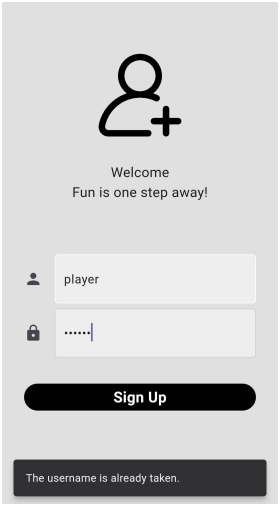


Figure 12: Sign-up Screen on Mobile

Figure 13: Leaderboard

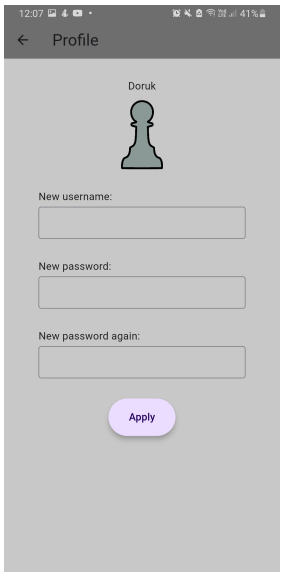


Figure 14: Login Page on Mobile

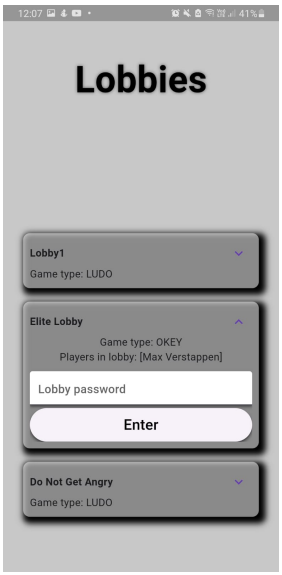


Figure 15: Lobby Screen on Mobile

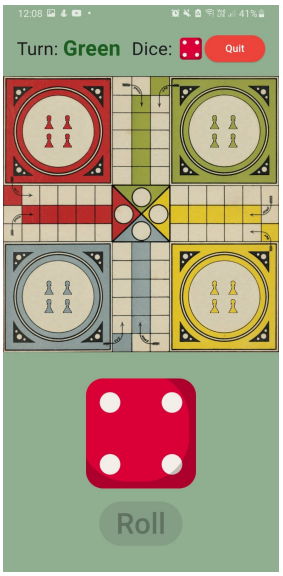
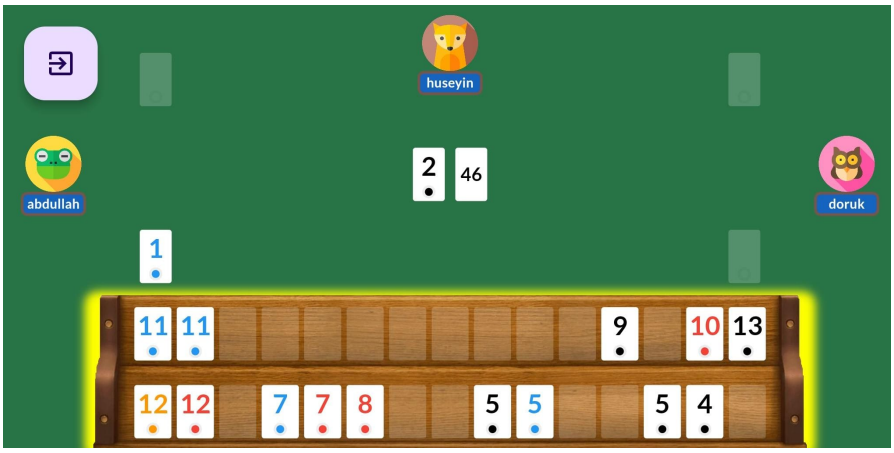
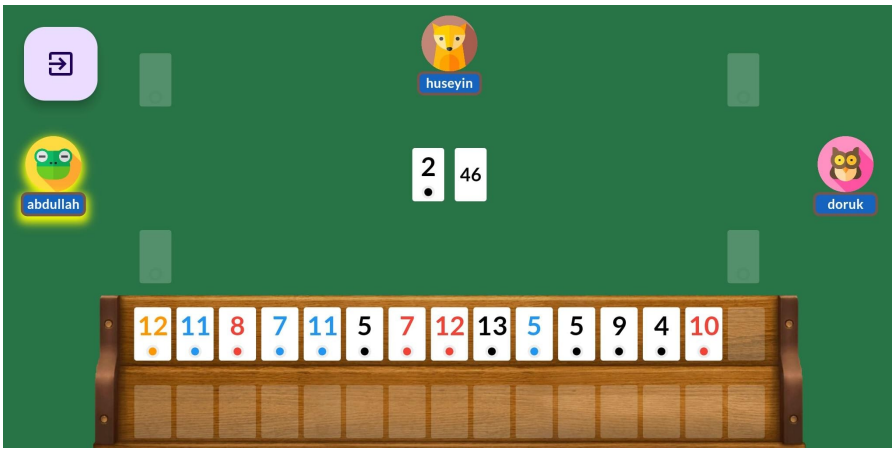


Figure 16: Ludo King Game on Mobile





7 Challenges and Solutions:

7.1 Challenges and Solutions for Server Module:

7.1.1 Automatic Server Startup:

- Challenge: Ensuring the server starts automatically upon Raspberry Pi power-up.
- Solution: Implemented a background service that triggers when the Raspberry Pi is powered on. The service waits for a Wi-Fi hotspot (PiPlayBox) to open and initiates the server accordingly.

7.1.2 Divergent Client Requirements:

- Challenge: Handling diverse needs from mobile and desktop clients within the same requests.
- Solution: Implemented a mechanism to identify the client type using session tokens. Adjusted responses based on the client type, ensuring compatibility and consistent communication.

7.2 Challenges and Solutions for Mobile Application Module:

7.2.1 Server Connection with Socket Implementations:

- Challenge: Establishing a reliable connection with the server using socket implementations.
- Solution: Utilized Flutter libraries to implement asynchronous TCP connection requests, resembling RESTful services. This approach provided an effective solution for seamless communication.

7.2.2 Complex Implementation for Multiple Clients in One Build:

- Challenge: Managing multiple clients within a single build led to complex implementation.
- Solution: Collaborated closely with the server, employed suitable checker functions, and utilized a username-based logic to facilitate the implementation for handling multiple clients efficiently.

7.2.3 Adapting Components to Different Devices and Screens:

- Challenge: Placing components dynamically on various devices and screens proved challenging.
- Solution: Implemented a scalable system where each component is placed individually, ensuring consistent layout across different devices and screen sizes.

7.2.4 Timer Logic and Turn-Based Implementations:

- Challenge: Implementing timer logic and turn-based features suitable for all users.
- Solution: Incorporated server responses, parsed and placed them correctly into checker variables, facilitating the successful implementation of timer logic and turn-based functionalities.

7.2.5 Okey Game Logic:

- Challenge: Developing the logic for the Okey game, involving tile interactions and user clicks.
- Solution: Implemented tile logic based on turn-based and click-based rules, aligning with the general logic of the Okey game.

7.2.6 Ludo Game Logic:

- Challenge: Handling the complexity of the Ludo game with 225 different tiles and scaled blocks.
- Solution: Employed hard-coded logic for each color of the pawns, ensuring smooth functionality and interaction with the game elements.

7.2.7 Communication Challenges in Lobby, Start, and Turn Logic:

- Challenge: Addressing communication issues in lobby, start, and turn logics.
- Solution: Implemented a responsive design and user highlights to enhance the app's clarity and ease of understanding.

7.2.8 Security Concerns on Android and iOS Devices:

- Challenge: Addressing security problems preventing server communication with Android and iOS devices.
- Solution: Managed security issues within the XML files of the Flutter application, implementing proper security measures to ensure seamless communication.

7.3 Challenges and Solutions for Desktop Application Module:

7.3.1 Preventing UI Blocking with Multithreading:

- Challenge: Frequent game requests to the server (5 times per second) were causing UI blocks.
- Solution: Introduced a new thread class for each game, preventing UI blocking and ensuring smooth communication with the server.

7.3.2 Communication Challenges between C++ and QML:

- Challenge: Issues arose in communication between C++ and QML, especially in areas requiring instant updates.
- Solution: Conducted research to identify areas requiring immediate changes and implemented QProperty logic to enhance communication between C++ and QML seamlessly.

7.3.3 Accumulation of QML Files on Stack during Transition:

- Challenge: Transitioning between QML files resulted in the accumulation of files on the stack, causing potential issues.
- Solution: Implemented a loader component in the main QML file to manage transitions effectively and prevent the stacking of QML files.

7.4 Challenges and Solutions for Virtualization Module:

7.4.1 Proper Placement of Pieces and Pawns:

- Challenge: Invisible pieces and pawns were not moving correctly to their designated areas, posing a challenge to proper placement.
- Solution: Implemented a strategy to place invisible pieces and pawns in every movable area, ensuring they moved correctly and occupied the expected positions.

7.4.2 Simultaneous Animation Display:

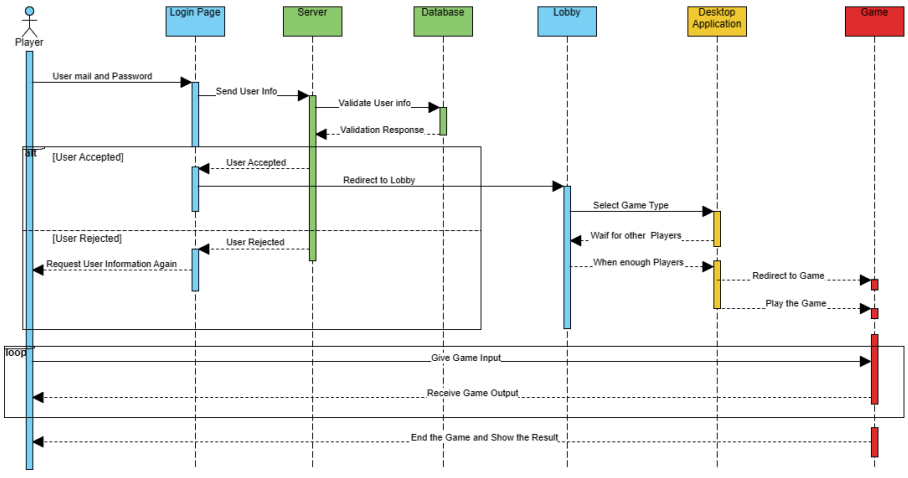
- Challenge: Displaying animations for movement (x, y, z, rotation) required a method to showcase them simultaneously.
- Solution: Utilized Parallel Animationproperty to ensure synchronized animations, creating a cohesive and visually appealing experience.

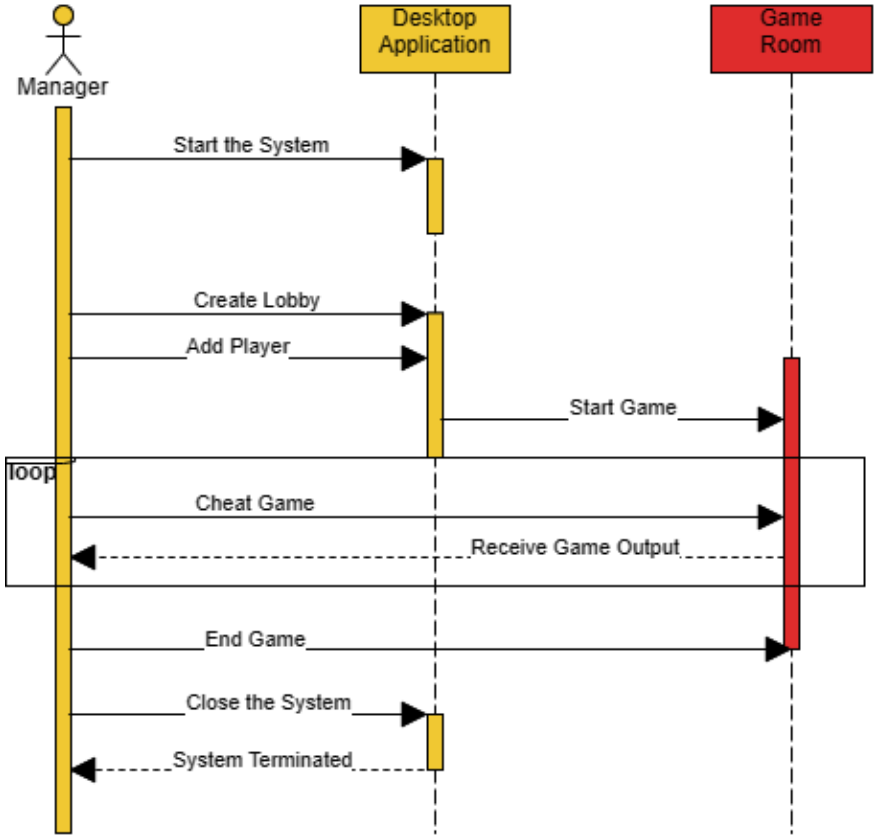
7.4.3 High CPU Consumption after Exiting Games:

- Challenge: Exiting games and returning to the game selection screen resulted in high CPU consumption due to lingering threads.
- Solution: Implemented a mechanism to safely close threads whenever the virtualization screen was closed, effectively reducing CPU consumption and improving efficiency.

8 Diagrams:

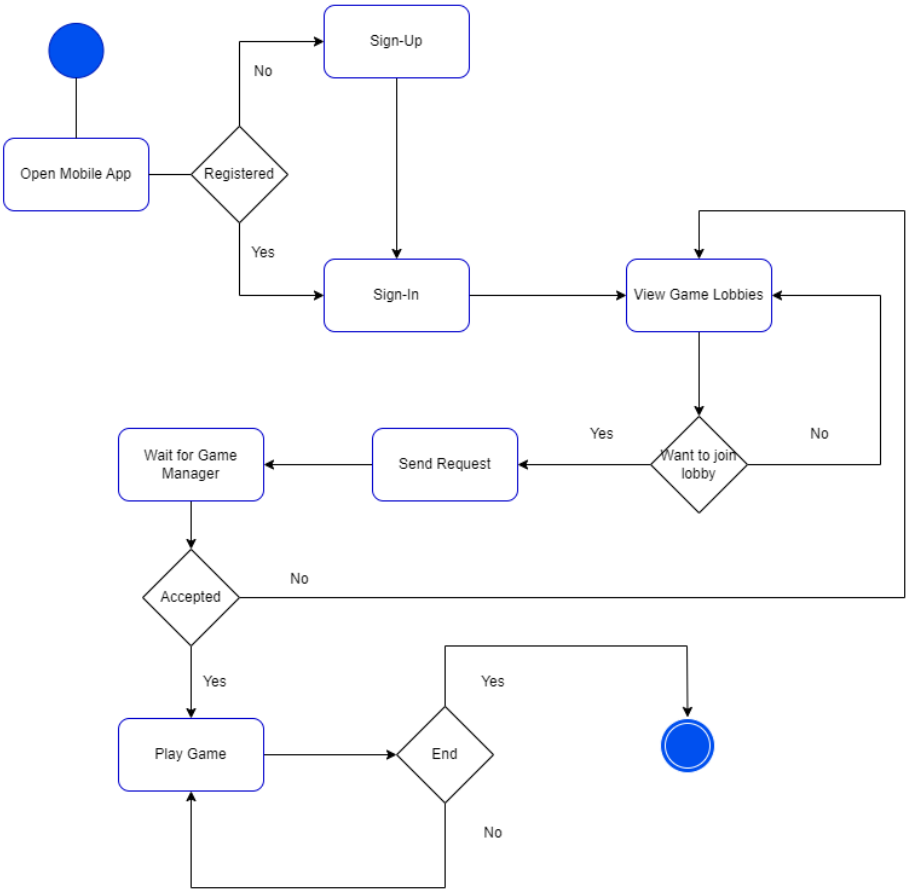
8.1 Sequence Diagrams:




Manager
Desktop Application
Game Room
Start the System
Create Lobby
Add Player
Start Game
loop
Cheat Game
Receive Game Output
End Game
Close the System
System Terminated

8.2 Process Diagrams:

Play Game Process for User



Play Game Process for Game Manager