CHAPTER 1
INTRODUCTION

1.1 PROJECT DESCRIPTION

The project I have developed is an online multiplayer first-person shooter game realized in the Unity engine with Photon PUN2 for networking. It is an FPS game intended to provide a captivating and competitive way for many players to connect and play in fluent free-for-all mode create or join a lobby and be able to include several players connected and playing against each other in real-time.

The core game mechanics include fast-paced shooting action in a game environment where players move around, devise strategies and try to outshoot their rivals. Photon PUN2 will help in smooth multiplayer interactions that can be undertaken by players. To that respect, Photon PUN2's networking will be very important in keeping the game in real-time so that players are able to see others are doing almost at the instance of the action.

This game developed in C# uses scripting in Unity for most of the game mechanics, including but not limited to player controls, shooting functionality, game state management and even creating lobbies.

This game is based on Photon PUN2, which provide the functionality of real-time multiplayer online game. The main development environment creating a great development environment would be based on Unity and C#. All of these have been applied correctly to make a game that has the potential to please from a playability perspective and also as proof of the technical abilities learnt during studies.

Besides the core mechanics and networking features described above, it contains several options for customization and settings that improve playing the game. Players are able to change the colour of the character before they start the game, making a different skin or colour in personalization gives no difficulty of identification in the game.

Furthermore, the audio system of this game is also amazing. The music in the main menu gives the energy to make us feel a warrior then during the matches, the SFX of the game also give us energy and some audios helps the player to understand what is happening around them. In main-menu players can controls the sound by dragging the slider to change the volume of various game sound into whatever the player wants to really get loud and excited during intense moments or very low.

Networking given by Photon PUN2 was used which is in real-time; thus, all players will definitely be on the same page. This level of game state management is necessary for events and actions to be reasonably vetted into a coherent manner for all connected players, improving in competitive and strategic nature of the game.

Combining of such features with developed networking by Photon PUN2 and immersive game mechanics makes for quite a full and engaging multiplayer FPS game. This project developed in Unity with C# shows not only technical knowledge acquired during studies but also the potential to create fun, competitive game, well-looking, that players will want to play.

1.2 COMPANY PROFILE

DxMinds Innovation Labs is accredited with the ISO 9001:2015 quality management system for developing incredible digital solutions giving a dynamic transformation to the digital world. Also, this company has been awarded as the start-up of the year 2020 by Silicon India for disrupting the traditional market trends and reshaping them with the drift of the latest technology.

DxMinds delivers end-to-end digital transformation solutions and technology services, ranging from ideation to designing and from development to deployment. For a decade this company has been serving the widest array of business niches. DxMinds is an agile studio that offers the most customized solutions across the digital value chain. This company is highly proficient in technologies like Artificial Intelligence, Chatbots, Mobile Applications, AR/VR, IoT, Web Apps, Blockchain, and more. Whether we are a start-up or a well-established organization willing to up-scale our business with an enthralling business app. This company offers unmatched services capable of giving cutthroat competition to the existing market with revolutionary technological change.

CHAPTER 2
LITERATURE SURVEY

2.1 Existing and Proposed system

2.1.1 Existing System

This online multiplayer first-person shooter game operates with different components that integrate smoothly to offer players an exciting game experience. Photon PUN2 is a real-time multiplayer networking engine developed especially for Unity.

Unity is a powerful and flexible game development platform capable of supporting 2D and 3D graphics, physics and scripting. This can provide design, development and testing of the game with a fully featured set of tools.

With the advanced inbuilt graphics engine and physics system, Unity eases the process of developing both beautiful and realistic game environments. Advanced features include dynamic lighting, detailed textures and realistic physics interactions that work to provide immersive game worlds. Photon PUN2 provides low latency and strong synchronization; without it an FPS game would be unable to sustain its fast phase.

Gameplay mechanics such as player movement, aiming, shooting mechanisms, health and damage systems and scoring are developed in C# using Unity's scripting capabilities. As such, the game logic ensures that all activities of the players are accounted for in the game world to yield a fair competitive experience. First-person view through which players control their characters: movement around the environment, weapon aiming and combat.

The user interface is very friendly, the lobbies creation and joining, the scores of the players, and game statistics display have been made easy. All this in UI design makes it much easier for players to get a game up and running fast, keep watch over progress and understand their performance.

It provides a ton of strong development tools available in Unity, networking capabilities with Photon PUN2 and flexibility in C# programming to deliver at one's fingertips a solid foundation for an entertaining and competitive multiplayer first-person shooter gaming experience.

2.1.2 Proposed System

One of the systems I envision for my online multiplayer first-person shooter game aims to extend the current framework in certain fundamental ways an interesting, competitive event of gameplay realization. Improvements will target areas such as networking, gameplay mechanics, user and game environments.

Improved Networking

The proposed system aims to optimize the networking infrastructure that enables multiplayer using Photon PUN2. This involves matchmaking, refining algorithms in parity matching players by skill levels, creating a much more balanced and enjoyable environment in gameplay. Furthermore, latency and desynchronization issues are expected to be kept to a minimum providing smoother gameplay.

Refined Gameplay Mechanics

The proposed system will bring a lot of refinements to the core gameplay mechanics including:

- Enhanced Character Movement: Refine the mechanics of character movement so that they are responsive, realistic and immersive. Each aspect related to character movement should allow players to feel perfectly immersed in their respective game worlds.

More Intuitive and Appealing User Interface

The user interface shall be revamped to have a more intuitive and appealing look. Critical improvements include:

- Simplified HUD: An enhanced heads-up display which clearly communicates all the necessary information without cluttering the screen.
- Improved User-Friendly Lobby System: Overhauling the lobby system for creating and joining games easily with clear options and highly recognizable controls.

Enhanced Game Environments

To improve the gaming environments in the title, the following will be done:

- Detailed Maps: More detailed and varied maps with strategic possibilities and challenges.

Basic Analytics and Monitoring

The basic analytics and monitoring integrated into the proposed system will track data related to player behaviour, the performance of games and network stability.

- Improving Game Balance: Adjust game mechanics based on feedback and performance metrics from players.
- Performance Optimization: Problems in the network are to be diagnosed and fixed keeping in mind the aims of minimizing latency and improving servers' performance.

Security Features

The proposed system will embed some basic measures of security so as to clean and transparent environment for gaming. This shall include:

- Regular Updates: Users shall be provided with regular updates for eliminating vulnerabilities and improving the security of the game in general.

The overall system to be proposed will aim at the betterment of the online multiplayer FPS game by introducing its main improvements and optimizing performance to improve the overall player experience. By capitalizing on the power of existing technologies and taking into consideration the feedback from players, the game will be more immersive, competitive and enjoyable for each player.

2.2 FEASIBLITY STUDY

The online multiplayer first-person shooter genre game project involves assessments in several totally different areas to determine whether it is viable. These are basically focused on technical, operational, economic and schedule feasibilities.

Technical Feasibility

1. Technology and Tools: The game is supposed to be developed using Unity as a game engine, while Photon PUN2 is used for networking. Both tools are well-established in the gaming industry therefore, reliable for use with provided support. C# shall be the main language of programming, building on top of Unity scripting capabilities.

2. Development Skills: I have enough knowledge of Unity, Photon PUN2 and C# programming. Familiarity with these technologies establish structured development and allows problem-solving abilities.

3. Infrastructure: Game project necessaries a strong server infrastructure to maintain real-time multiplayer interactions. Photon PUN2 provides cloud services that can be scale, therefore it will efficiently handle the players that may be expected, providing smoothness and responsiveness gameplay.

Operational Feasibility

1. User Requirements: Most of the FPS gamers are demanding competitive gameplay, smooth and convenient multiplayer controls and incredible game environments.

2. Maintenance and Support: Our project planning includes regular updates and maintenance to fix the coming bugs, adding new exciting features and improve the gameplay.

Economic Feasibility

1. Development Costs: Unity and Photon PUN2 are free and we can get and access the features which they provided, but if we want to use some more advance and more functionalities of these technologies, we have to pay to them depends on what we use.

2. Generate Revenue: We could use in-app purchases, ads, selling character skins or sell a premium version of the application to monetize the game.

Schedule Feasibility

1. Project Roadmap: The project is planning to involve a detailed time frame of all the aim for each phase of development, starting from the initial design to final testing. Fixing deadlines of the project at each stage so we can easily keep on the track.

2. Resource Allocation: Proper resources are available to each part of the project together with the provision of the necessary tools and software that would be needed for development. In this way, developmental tasks will be finished efficiently within the estimated timeframe.

3. Risk Management: The technological, budgetary and schedule risks that may potentially blow up during the course of the project are envisaged in advance, mitigation

measures and strategies have been adopted and regular progress reviews shall resolve the issues as and when they crop up.

Conclusion

The feasibility study shows this online multiplayer FPS game project to be fully viable on all main counts. The tech base is sound with dependable tools and personal experience. Operationally, the game fulfils user who-have-been requirements with provisions for maintenance and support of the game post-launch. Economically, the budget for the project is feasible while having very good revenue potential. The timeline for the project is realistic and features the allocation of adequate resources with risk management strategies in place. Meaning, this project is assessed as feasible and likely to succeed based on these factors.

2.3 Technology and Tools

Developing the online multiplayer FPS game shall base on the formidable set of technologies and tools to provide seamless, engaging and competitive game experience. The components are:

Unity Engine:

- Game Development Platform: Unity is a powerful and flexible game development platform capable of supporting 2D and 3D graphics, physics and scripting. This can provide design, development and testing of the game with a fully featured set of tools.
- Graphics and Physiques: With the advanced inbuilt graphics engine and physics system, Unity eases the process of developing both beautiful and realistic game environments. Advanced features include dynamic lighting, detailed textures and realistic physics interactions that work to provide immersive game worlds.

- Cross-Platform Support: Unity supports deployment on many platforms. It includes windows, macOS, Android, iOS and consoles. This makes the game available to a much larger audience.

Photon PUN2: Networking Solution: Photon PUN2 is a real-time multiplayer networking engine developed especially for Unity. It provides Stamper with an infrastructure that consider the seamless inbuilt multiplayer interactions.

- Scalability: Photon PUN2 has scalable cloud services that run different loads, hence ensuring smooth game performance with increased players.
- Real-Time Communication: Photon PUN2 provides low latency and strong synchronization; without it an FPS game would be unable to sustain its fast phase. All player connections and state updates are efficiently handled with real-time data transmissions.

This is a very resilient networking solution for Unity's online multiplayer games. It ensures smooth gameplay experiences through handling of player connections, state updates and real-time data transmissions. Scalability is one of its major features. This means that it can back up any load in the cloud so as not to decrease your game performance if you increase the number of players. Therefore, scalability is very important for high quality with fast FPS where most players play together concurrently.

Another critical factor is real-time communication in Photon PUN2. For an FPS game, low latency and high synchronization must keep pace with the rapid pace of action. Without these features, it would be difficult to maintain fluidity and responsiveness that are necessary for competitive multiplayer games. All player connections and their states update in Photon PUN2 are managed efficiently so that actions and events are always reflected on each player screen at any one time. Such low latency communication creates immersive gaming experience and fairness in a game allowing players to nearly instantly respond to each other's moves.

Other benefit features also supported by Photon PUN2 including matchmaking, lobby management and game communication. These can create a thorough multiplayer experience so that players can connect, interact and effectively contest or compete with others. Therefore, the game will bring a smooth, satisfying and competitive environment. This makes flexible to the needs of the players through extensive networking solutions given by Photon PUN2.

C# Programming Language:

- Scripting: This is the fundamental language for programming in Unity and this is mainly used to script game mechanics and controls with other game logic. The integration with Unity will help to make powerful, flexible code.
- Object-oriented: This object-oriented nature of C# is very useful for writing modular and reusable code, which is necessary to manage the complexity of a multiplayer FPS game.

Visual Studio:

- Integrated development Environment (IDE): Visual Studio is used as the IDE to write, debug and test the C# code. It has dynamic tools on editing, debugging and performance profiling of code.
- Unity Integration: Visual Studio has support in Unity including IntelliSense, Debugging, project and many others that will no doubt make development easier.

Photon Dashboard:

- Server Management: Photon Dashboard embeds tools to manage Photon servers allows performance monitoring of servers and extracts information about player activities. It answers questions about server performance and provides a stable multiplayer environment.
- Analytics: Tools for data collection and profile gathering on player behaviour and the performance of the game and network, which help greatly during decisions on restitution and game balance.

Blender:

- 3D Modelling: Blender is used for preparing 3D models, textures and animations for the game. It represents one of the great open-source tools that offer complete functionality for creating assets.
- Asset Export: More Blender and Unity compatibility ensures that 3D assets are easily exportable into the game, making sure it always shows high-quality visuals and performance.

Git:

- Version Control: Git is used to control the version, to keep tracking the codes of the game and associated assets. This allows following, collaborating and managing versions of the project.
- Backup and Collaboration: Git ensures that the project is backed up and changes can if needed, be reverted. It allows for transparency of change, hence collaboration even though the project has been developed solo.

All these technologies and tools are co-opted in running the smooth development of an online multiplayer FPS game for a quality and fun gaming experience.

2.4 HARDWARE AND SOFTWARE REQUIREMENTS

Hardware requirements

• Computer Processor	: Intel Core i5 or AMD equivalent (Quad-core or higher)
• RAM	: 8GB or more
• Graphic Card	: NVIDIA GeForce GTX 1060 or AMD Radeon RX 580
• Monitor	: Full HD (1920x1080) resolution
• Storage	: SSD 500GB
• Peripherals	: Keyboard, Mouse and optional game controller

Software requirements

• Unity Engine	: Latest Version
• Unity Hub	: for managing Unity version & project
• Photon Pun2	: SDK for Unity
• Programming Tools	: Visual Studio or later
Source Control	: Git (with platform like GitHub or GitLab)
• 3D Modelling	: Blender (latest stable version)
• Image Editing	: Adobe Photoshop

CHAPTER 3
SOFTWARE REQUIREMENT SPECIFICATION (SRS)

Purpose

This document is purposed to define and lay down the constraints on the software requirements in the implementation of an online multiplayer first-person shooting game. This document also gives detailed explanations on the functional requirement, non-functional requirement, interfaces sought and the constraints of the system.

Scope

The present SRS document describes the elaboration process for an FPS game in Unity with Photon PUN2 and is expected to include the creation of lobbies, search for games and join and finally a free-for-all mode.

Table 3.1.1 Definitions, Acronyms, and Abbreviations

FPS	First-Person Shooter
SRS	Software Requirements Specification
Unity	A cross-platform game engine
Photon PUN2	Photon Unity Networking-2, Unity networking framework
HUD	Heads-Up Display

Overall Description

Product Perspective

The FPS game is constructed as a standalone project with the use of Unity and Photon PUN2. The game will ensure smooth multiplayer gaming and real-time communication among players.

Table 3.1.2 Product Functions

Creation of lobby	A player is able to create and handle created lobbies.
Joining of Lobby	Easily join previously formed game lobbies by searching.
Free-for-All Gameplay	Allows players to fight other players, it's a free-for-all type of game.
Player Controls	Move, aim, shoot and use abilities.
Scoring System	It notes the player's high score and ranks players accordingly.

Character Customization	With different skins and gears, player personalization will be possible.
Game Sound	Includes immersive in-game sound effects and music, with options to adjust volume settings.
Setting Game Duration	Players can set the game duration before the game starts, ensuring matches have defined time limits.
Social Features	Contains friends lists and in-game chat functionality.

Assumptions and Dependencies

- The multiplayers game component is reliant on an internet connection that is stable.
- The end users possess hardware with minimal specifications.

SPECIFIC REQUIREMENTS

Functional Requirements

This is an expanded and more detailed version of the functional requirements:

Lobby Management

- Creation of Game Lobbies: Provide system functionality so that players are able to create game lobbies with a private setting in which they could set game parameters, invite friends to join in a game and manage player participation. This includes naming the lobby, defining a maximum number of players and setting game rules.
- Joining a created game lobby: Players will join an existing game lobby by searching all the available lobbies and then joining them. The system shall provide smooth and easy joining of any lobby, where it will clearly show whether a lobby is ready to start or if there are constraints on it.
- Available Game Lobbies Indication: The system will indicate to the player the available game lobbies, together with their lobby name, host, number of players and current game settings. This feature confirms that players have ease in finding and entering a game that fits their interests.

Gameplay

- Player Controls: The game shall be enabled so that players can control the characters easily and seamlessly move in all directions, change weapons and shoot. The controls should be intuitive and responsive to ensure an even gameplay flow.
- Score Display and Updates: A display of the players' scores is always shown on the screen and will update as a player proceeds to score. Scores are ever clear and readable. This will enhance the competitive nature of the game since players can always see how players are performing against their opponent.
- Game Termination and Result Presentation: If the game times out or reaches a certain score limit set for any player, the system will automatically terminate the game and present the results to the players. The results screen will include player rankings and scores, among other relevant statistics, thus summarizing the game.

Game Sound

The game delivers great in-game sound effects and a dynamized score adding to the atmosphere. Also, audio personalization is available through an in-game menu that allows volume settings for sound effects and music to be changed according to taste. This means it provides a variable, friendly auditory environment with regard to the different necessities of the players and increases its immersive capacity.

Social Features

- Friends List Functionality: Be able to add friends and maintain a list of them. Any player shall be capable of sending and accepting friend requests, getting online status of their friends and hence easily able to invite friends into their game lobbies.

It will have in-game chat functionality, enabling real-time communication among players. Such functionality shall support text chat and voice chat, thus enhancing the social features of the game and letting players correlate with each other more effectively in the game.

Customization

- Character Customization Options: Include a comprehensive character customization feature that will permit players to design their characters with a bunch of in-game skins, gears and other cosmetics.

- Customization Settings Maintenance: The system retains all of the customization settings that were set by the player across the game sessions. If a player has customized his or her character, the preferences get saved and will be applied automatically when one logs into the next gaming session. This thus makes up a coherent, tailor-made gaming experience.

Non-Functional Requirements

Performance Requirements

- The system should run with at least 30 FPS on all supported devices.
- The maximum latency doesn't exceed 100ms during the sessions.

Usability Requirements

- The interface should be easy to use and understand.
- In-game guidelines must be there to helping the new players and to understand how they can play as professionals.

Interface Requirements

Table 3.1.3 User Interfaces

Main Menu	The main menu is where the player can start a game or join a lobby, create a lobby, customize characters, view leaderboards, access settings and exit the game.
Join Lobby UI	In this interface, the player will be shown the list of open lobbies to which he/she can join. The interface provides options to filter and select lobbies based on filters like ranked, custom, etc.
Create Lobby UI	The Create Lobby UI allows players to create and be hosts to a new game lobby. Through this UI, players would be able to set the name of the lobby, the maximum number of players and other settings specific to the lobby.
In-Game UI	The In-Game User Interface displays the essential information throughout the game round, like the players' health and score, remaining time of the game, quantity of ammunition, etc. Other useful elements are included for entering the settings of the game & leaving the game.

Leave Game UI	The Leave Game UI provides the player with the opportunity to leave the current game session.
View Leaderboards UI	The View Leaderboards UI displays rankings and scores for players. It provides information about top scores and personal rankings and therefore compares each player with others in the situation of a game.

Hardware Interfaces

- Keyboard and Mouse: The system of player input on a PC.

Software Interfaces

- Photon PUN2 API: In charge of Networking and multiplayer interactions
- Unity Engine: Develops and renders the game.

Communication Interfaces

- Internet Connection: Needed for multiplayer functions and real-time transmission and reception.

Other Requirements

Legal and Regulatory Requirements

The system shall support all legal regulations concerning the protection of data, such as the General data protection for the privacy and security of users' information. This shall include the development of solid encryption techniques, secure storage and access control over personal information. Clear information should also provide to the users with respect to practices relating to collecting the data and an opt-out possibility where feasible, also access to their personal data under demand

Documentation Requirements

The document will be fully and maximally documented to back up every bit of the system. This will include detailed guidelines on how to install the system, thus making it easy for users to set up the game environment on different platforms. This will also contain comprehensive user manuals that would help all the players go through all the different attributes of the game, controls and settings so that bring out the best possible user experience. This will be followed by troubleshooting guides assisting the user in solving some of the common problems and errors on their own.

Support and Maintenance Requirements

Provisions in the system design should have provisions that include periodic update and maintenance activities to ensure the system's continuous operation, security and user experience. This will involve the development of an update mechanism that allows rolling out patches, new features and improvements without causing glitches in the user's experience. Schedules for regular maintenance will be put in place to ensure prompt dealings with bugs, performance issues and security vulnerabilities.

CHAPTER 4
SYSTEM DESIGN

4.1 Context Diagram

Understand the high-level data flow with other external identities

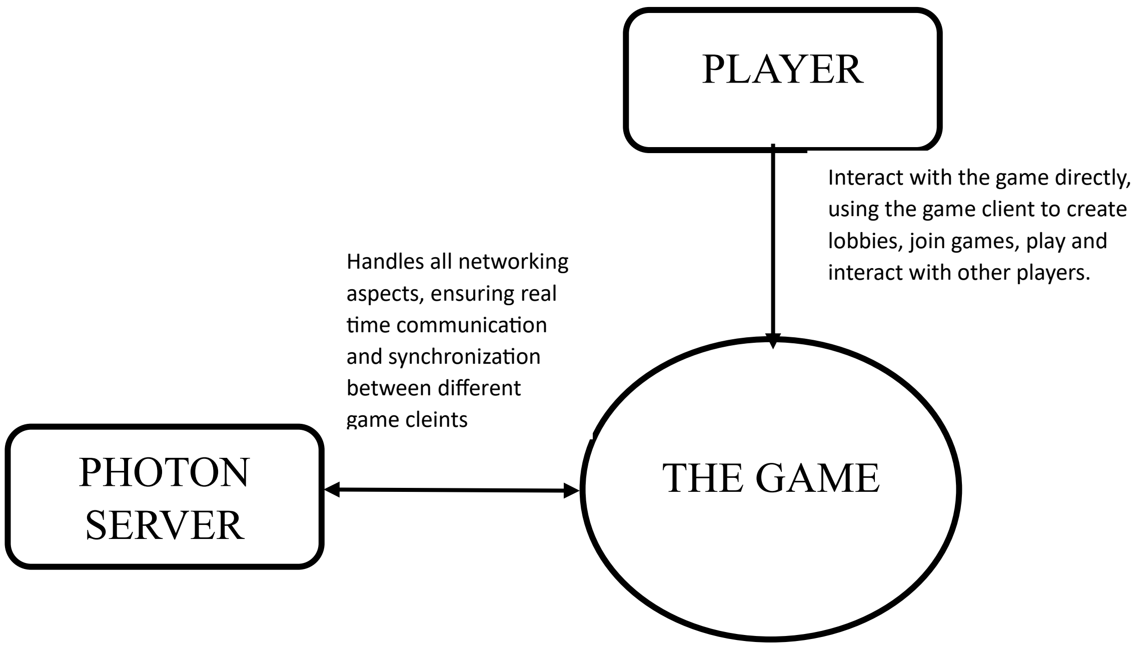
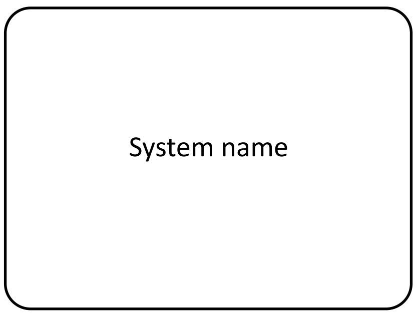

PLAYER
Interact with the game directly, using the game client to create lobbies, join games, play and interact with other players.
Handles all networking aspects, ensuring real time communication and synchronization between different game cleints
PHOTON SERVER
THE GAME

Figure:4.1.1 Context diagram

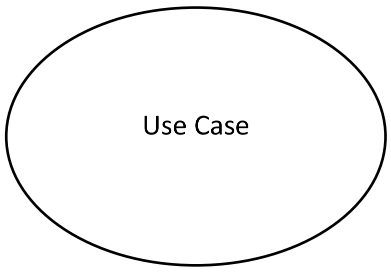
CHAPTER 5
DETAILED DESIGN

Use Case diagram

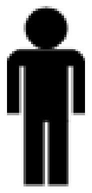
Links many details for reference on how users interact with the system.



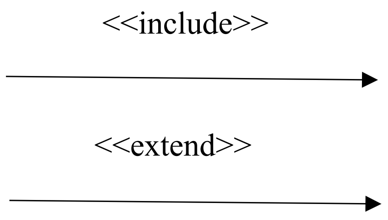
System



Use Case



Actor



Relationships

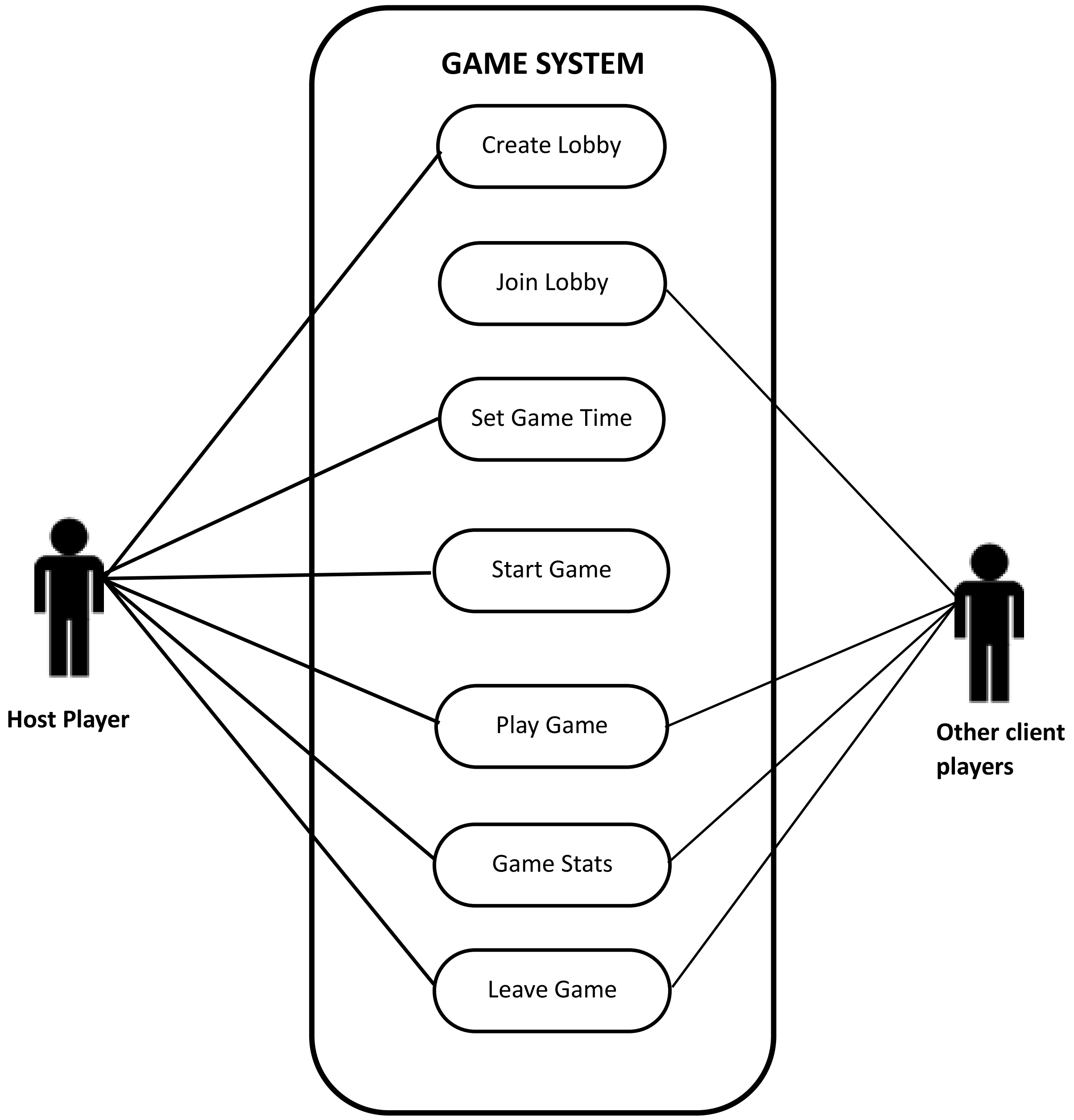


Figure:5.1.2 Use Case Diagram

5.2 Architecture Diagram

Help the colleagues to grasp the key idea thoughts of the framework

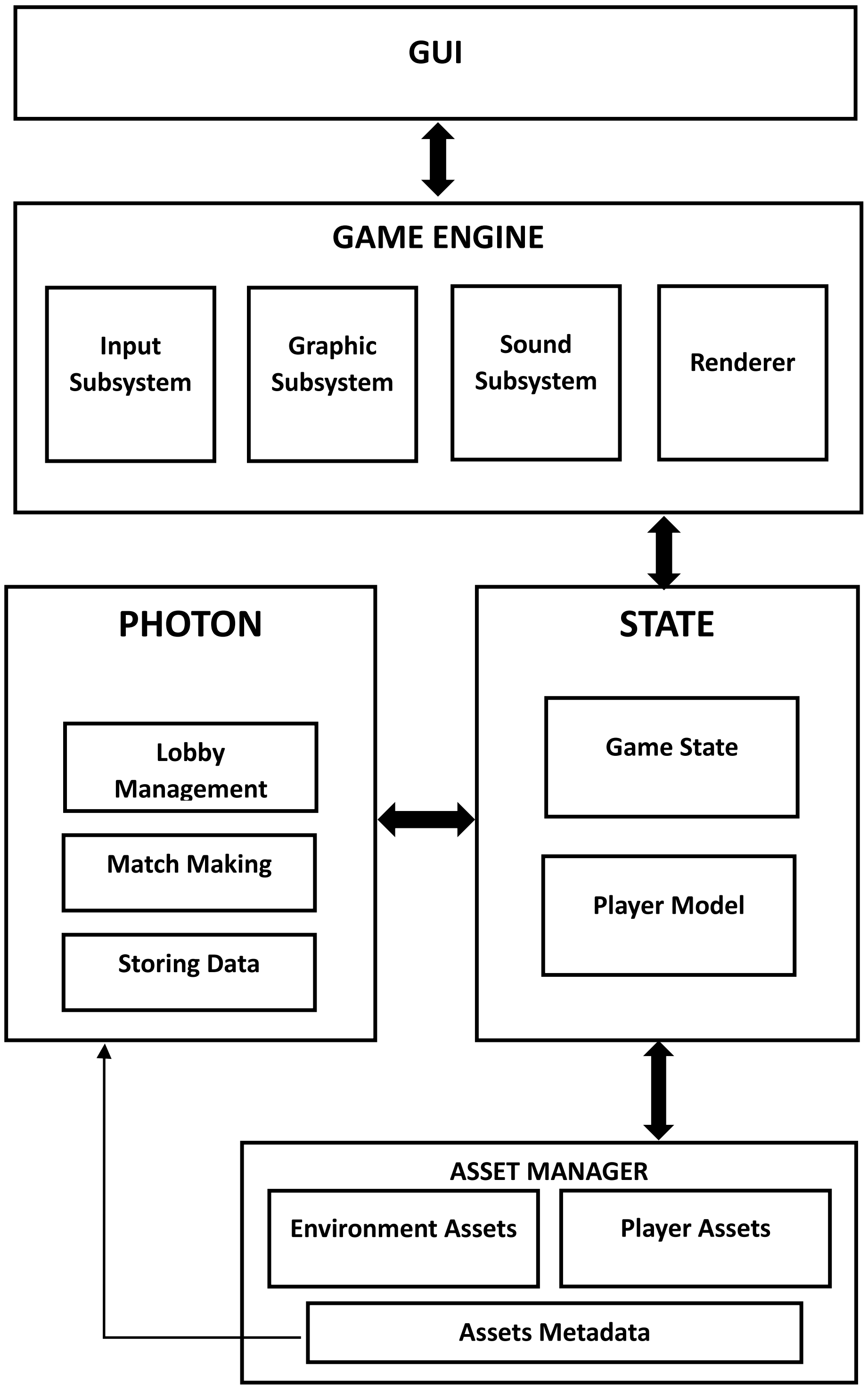
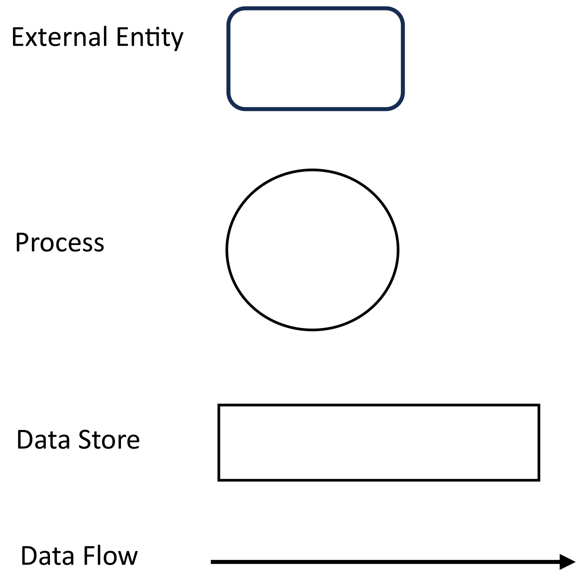


Figure:5.2.1 Architecture diagram

5.3 Data flow diagrams

Help us to understand the flow of connected processes and information



0-LEVEL DFD

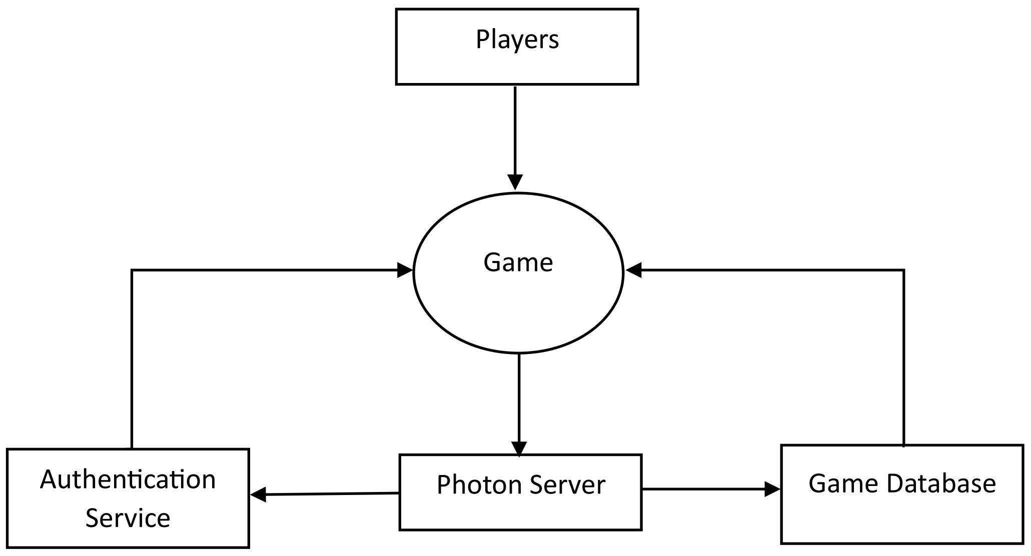


Figure:5.3.1 DFD Level 0

1-LEVEL DFD

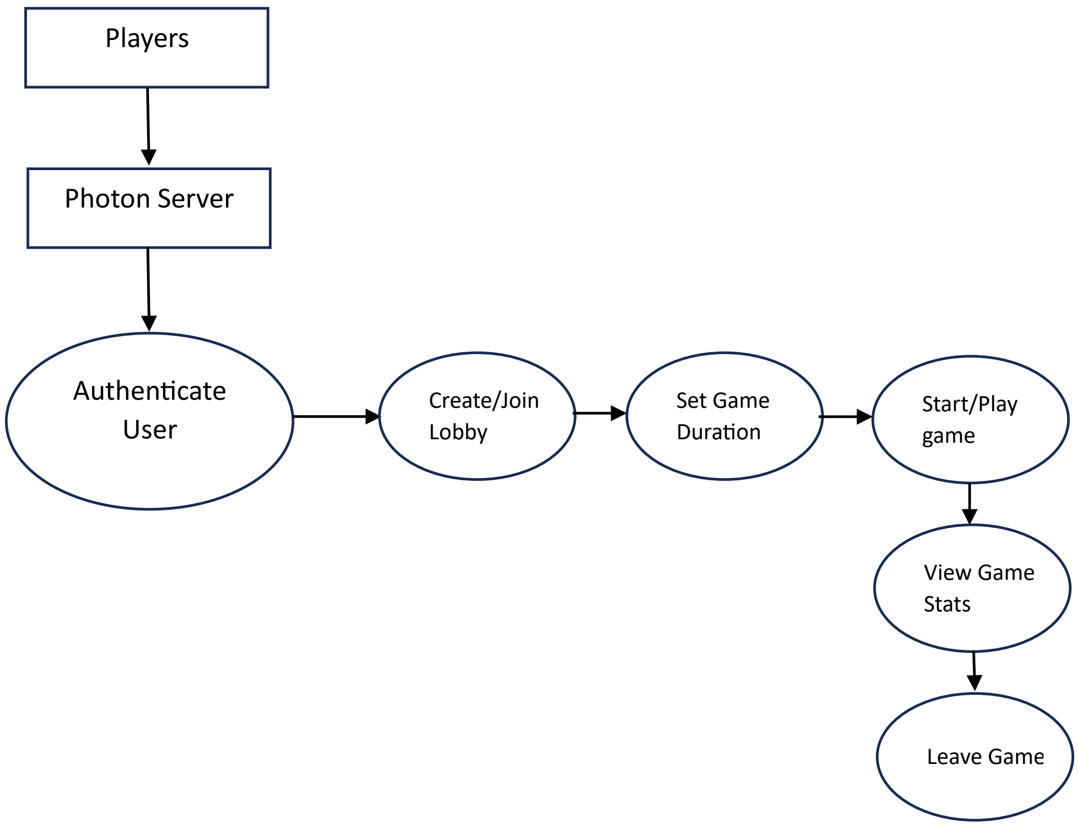


Figure:5.3.2 DFD Level 1

5.4 Sequence Diagram

To understand events generated by external actors

actor: Actor

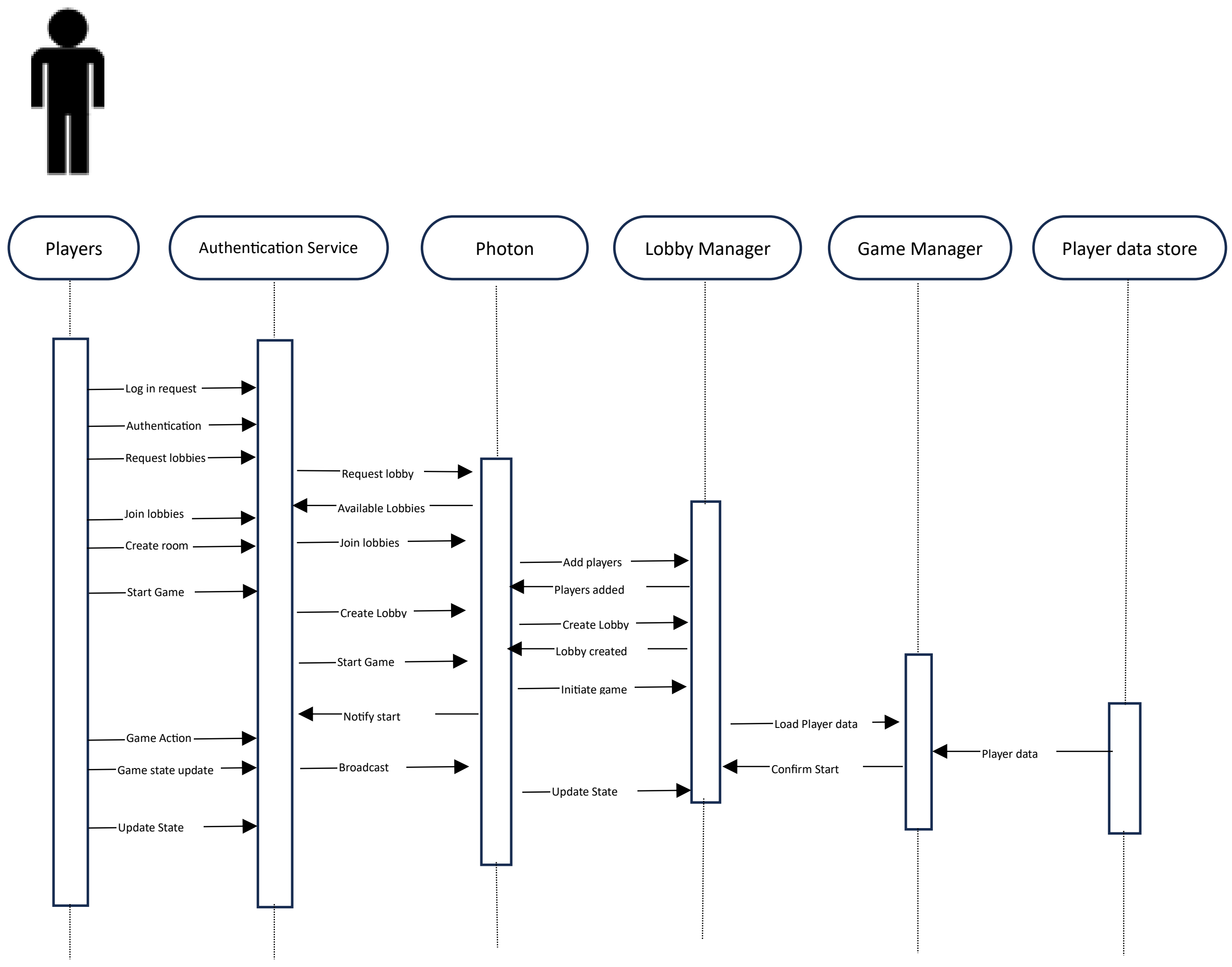
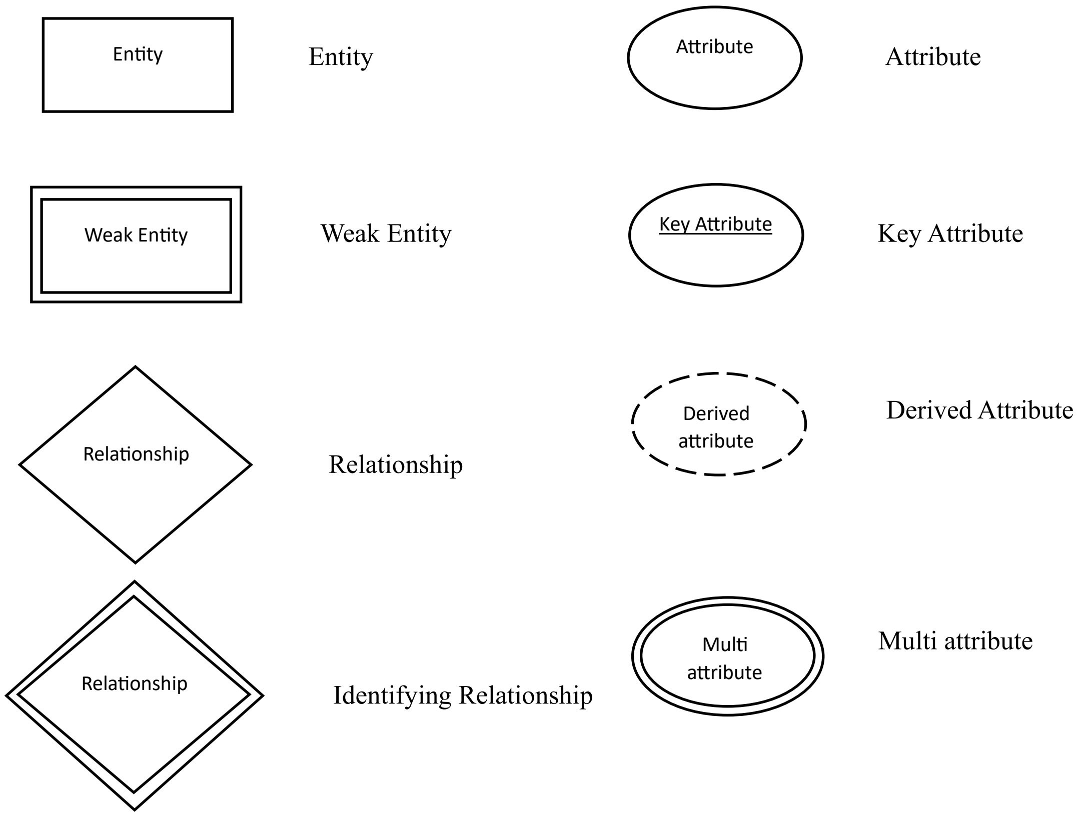


Figure: 5.4 Sequence diagram

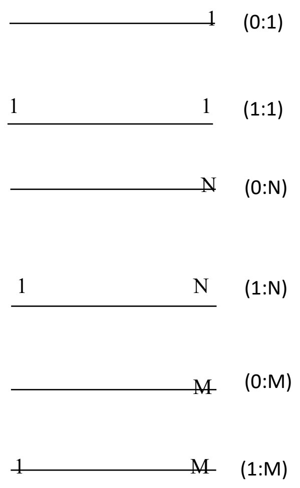
5.5 Entity relationship model

An entity relationship shows the relationships of elements within an entity

Chen's notation



Participations
Cardinality can be shown or hidden



Recursive Relationship
Cardinality can be shown or hidden

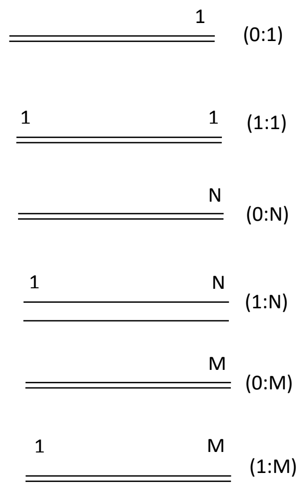


Figure: 5.5.1 Entity relationship

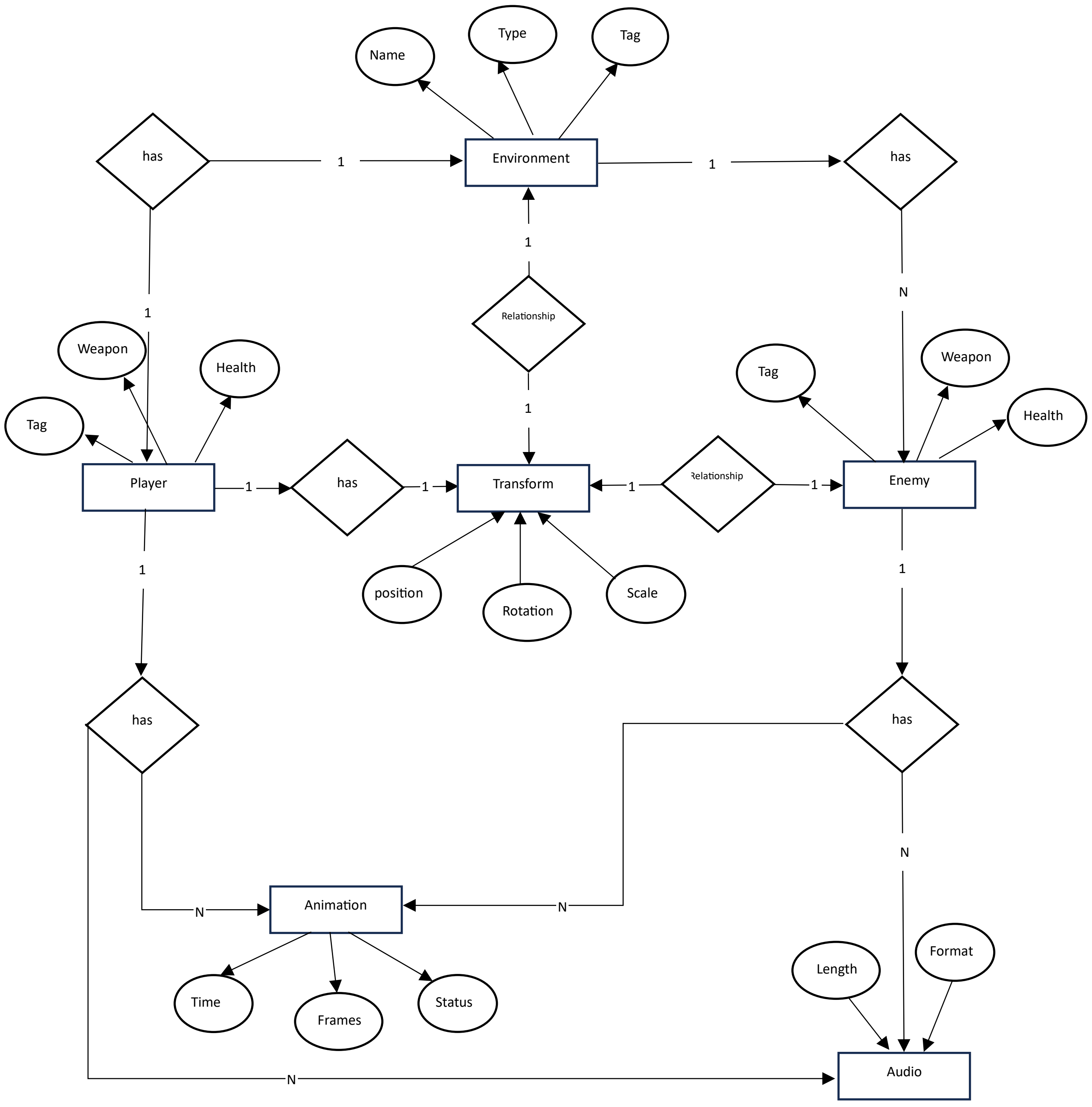


Figure: 5.5.2 ER diagram

5.6 Class diagram

To see every component and their connections

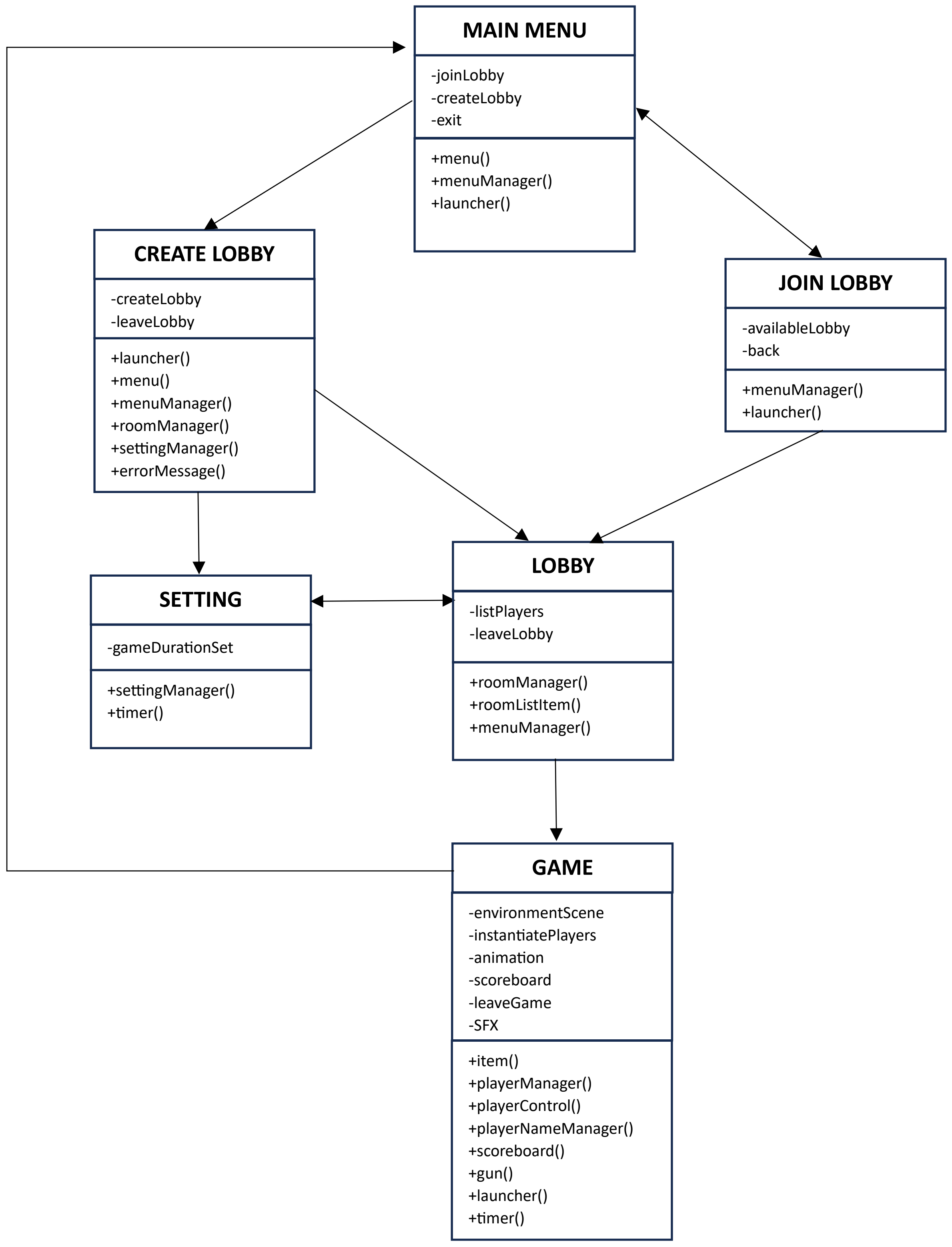


Figure: 5.6.1 Class Diagram

5.7 Activity diagram

A general workflow control is displayed for a specific activity

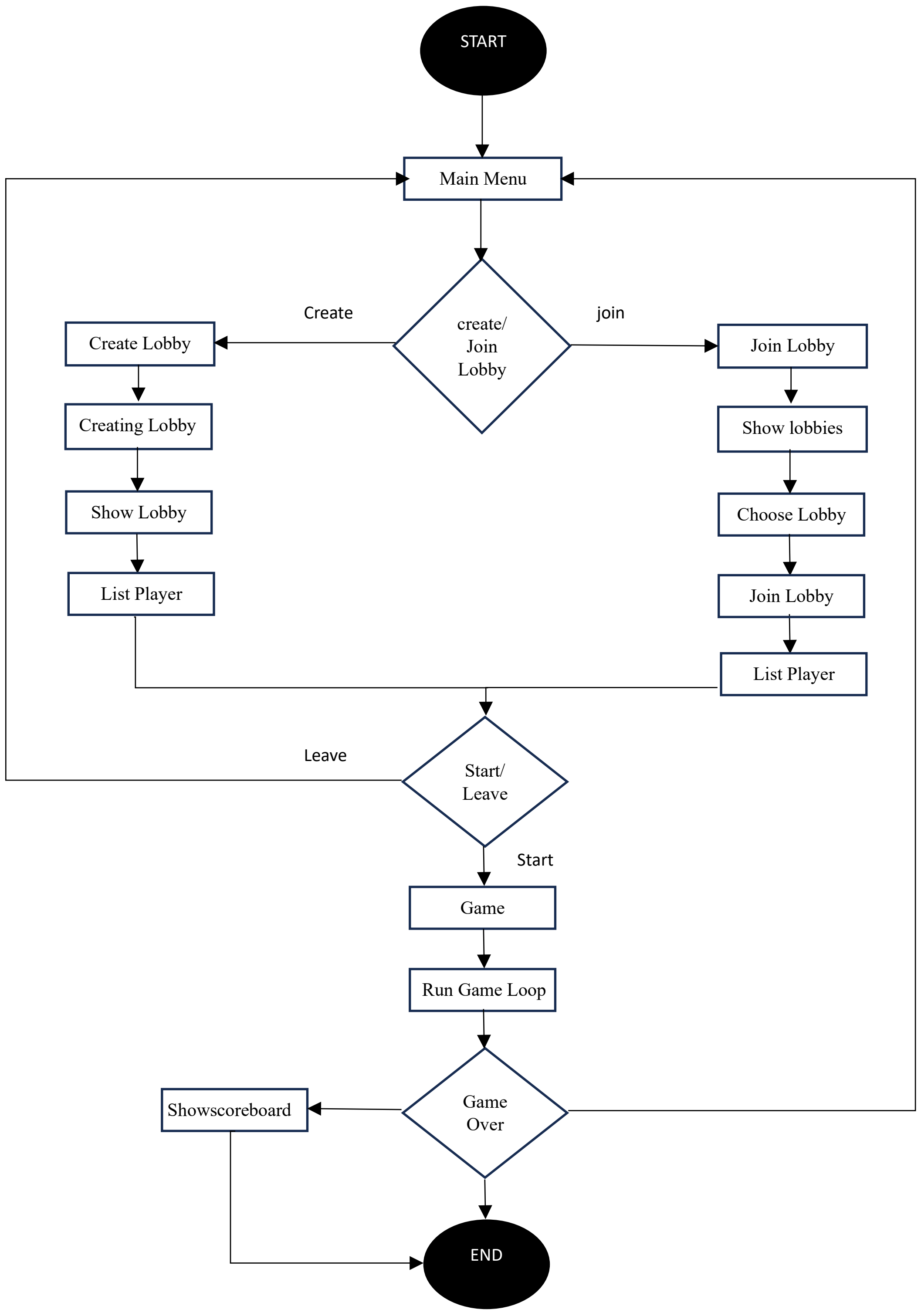


Figure: 5.7.1 Activity diagram

CHAPTER 5
IMPLEMENTATION

6.1 CODING

Launcher.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using Photon.Pun;
using TMPro;
using Photon.Realtime;
using UnityEngine.UIElements;
using System.Linq;
public class Launcher : MonoBehaviourPunCallbacks
{
    public static Launcher Instance;
    [SerializeField] TMP_InputField roomNameInputField;
    [SerializeField] TMP_Text errorText;
    [SerializeField] TMP_Text roomNameText;
    [SerializeField] Transform roomListContent;
    [SerializeField] Transform playerListContent;
    [SerializeField] GameObject roomListItemPrefab;
    [SerializeField] GameObject playerListItemPrefab;
    [SerializeField] GameObject startGameButton
    [SerializeField] GameObject settingGameButton;
    [SerializeField] SettingsManager settingsManager;



void Awake() {
        Instance=this;
    }
    // Start is called before the first frame update
    void Start()
```

```
  {
    Debug.Log("Connected to Server.");
    PhotonNetwork.ConnectUsingSettings();
    if (settingsManager == null){
    settingsManager = FindObjectOfType<SettingsManager>();
    }
    if (settingsManager == null){
    Debug.LogError("SettingsManager not found.");
  }
  else
  {
    Debug.Log("SettingsManager found.");
    settingsManager.gameObject.SetActive(false);
  }
  public  override void OnConnectedToMaster(){
    Debug.Log("Connected to Server.");
    PhotonNetwork.JoinLobby();
    PhotonNetwork.AutomaticallySyncScene=true;//automatically load the scene for all of
our client when the host switch the scene
  }
  public override void OnJoinedLobby()
  {
    MenuManager.Instance.OpenMenu("Title");
    Debug.Log("Joined Lobby.");
  }
  public void CreateRoom()
  {
    if(string.IsNullOrEmpty(roomNameInputField.text)){
      return;
    }
    PhotonNetwork.CreateRoom(roomNameInputField.text);
    MenuManager.Instance.OpenMenu("Loading");
  }
```

```
    public override void OnJoinedRoom(){// this gonna open the room menu and update the room name text
        MenuManager.Instance.OpenMenu("room");
        roomNameText.text=PhotonNetwork.CurrentRoom.Name;//to show the room name in room menu
        Player[] players=PhotonNetwork.PlayerList;
        foreach(Transform child in playerListContent){
            Destroy(child.gameObject);
        }
        for(int i=0;i<players.Count();i++)
        {
            Instantiate(playerListItemPrefab, playerListContent).GetComponent<PlayerListItem>().SetUp(players[i]);
        }
        startGameButton.SetActive(PhotonNetwork.IsMasterClient);//only host player can start the game
        settingGameButton.SetActive(PhotonNetwork.IsMasterClient);
    }
    public override void OnMasterClientSwitched(Player newMasterClient)// to switch the host
    {
        startGameButton.SetActive(PhotonNetwork.IsMasterClient);
        settingGameButton.SetActive(PhotonNetwork.IsMasterClient);
    }
    public override void OnCreateRoomFailed(short returnCode, string message){
        errorText.text="Room Creation Failed: "+message;
        MenuManager.Instance.OpenMenu("error");
    }
    public void StartGame(){
        PhotonNetwork.LoadLevel(3);//we use 3 as the parameter bcoz 3 is the build index of our game scene, as we set in the build setting
    }
    public void LeaveRoom(){
        PhotonNetwork.LeaveRoom();
        MenuManager.Instance.OpenMenu("Loading");
    }
    public void JoinRoom(RoomInfo info){
        PhotonNetwork.JoinRoom(info.Name);
        MenuManager.Instance.OpenMenu("Loading");
    }
    public override void OnLeftRoom(){
        MenuManager.Instance.OpenMenu("Title");
        PlayerPrefs.SetString("username", PhotonNetwork.NickName);//13
    }
    public override  void OnRoomListUpdate(List<RoomInfo> roomList){
        foreach(Transform trans in roomListContent){
            Destroy(trans.gameObject);// clear the list every time we get updated
        }
        for(int i=0;i<roomList.Count;i++){
            if(roomList[i].RemovedFromList)
                continue;
            Instantiate(roomListItemPrefab,
roomListContent).GetComponent<RoomListItem>().SetUp(roomList[i]);
        }
    }
    public override void OnPlayerEnteredRoom(Player newPlayer)
    {
        Instantiate(playerListItemPrefab,
playerListContent).GetComponent<PlayerListItem>().SetUp(newPlayer);
    }
    public void OpenSettings()
    {
        Debug.Log("OpenSettings called");
        if (PhotonNetwork.IsMasterClient)
        {
            if (settingsManager != null)
            {
                Debug.Log("SettingsManager is not null, calling ToggleSettings.");
                settingsManager.ToggleSettings();
                Debug.Log("SettingsManager.ToggleSettings() called successfully.");
            }
            else
            {
                Debug.LogError("SettingsManager is null, cannot call ToggleSettings.");
            }
        }
        else
        {
            Debug.LogError("Current client is not the MasterClient, cannot open settings.");
        }
    }
}
```

6.2 SCREENSHOT

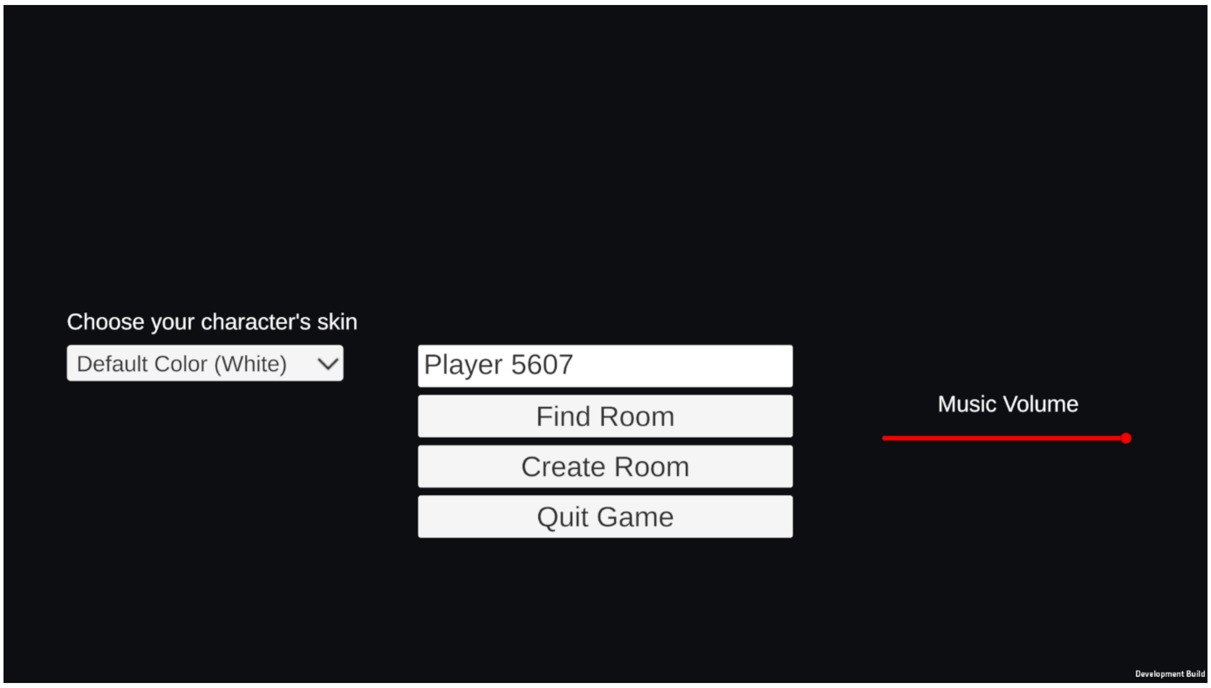


Fig. 6.1.1 Main Menu

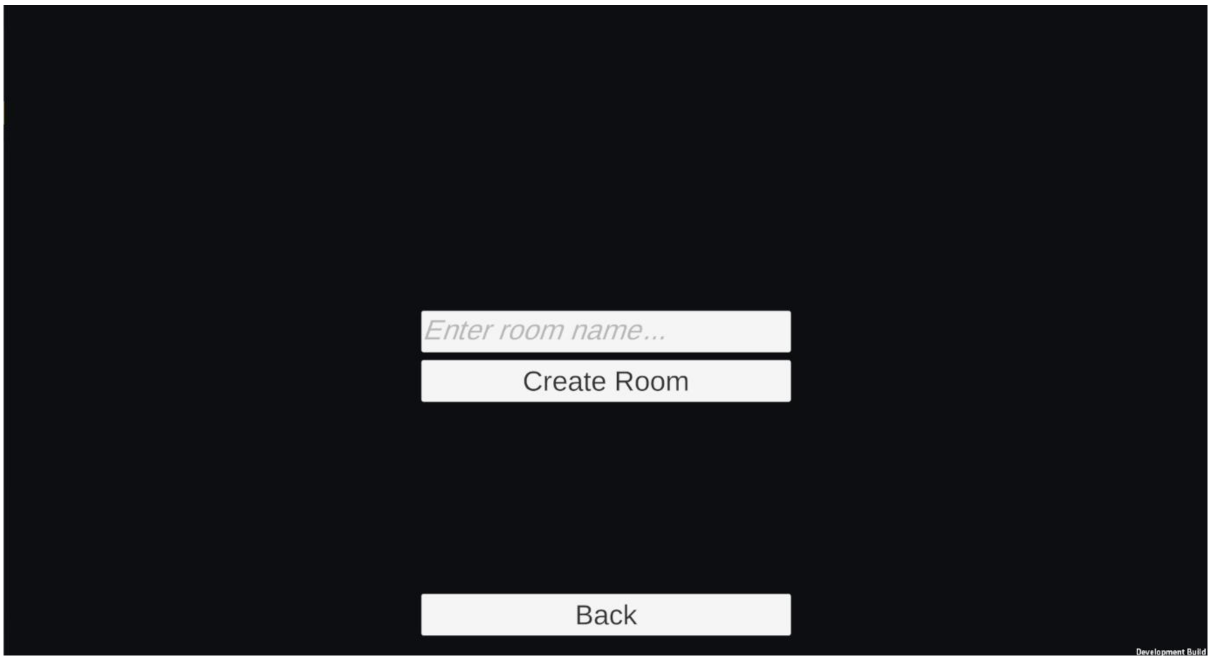


Fig. 6.1.2 Create-Room Menu

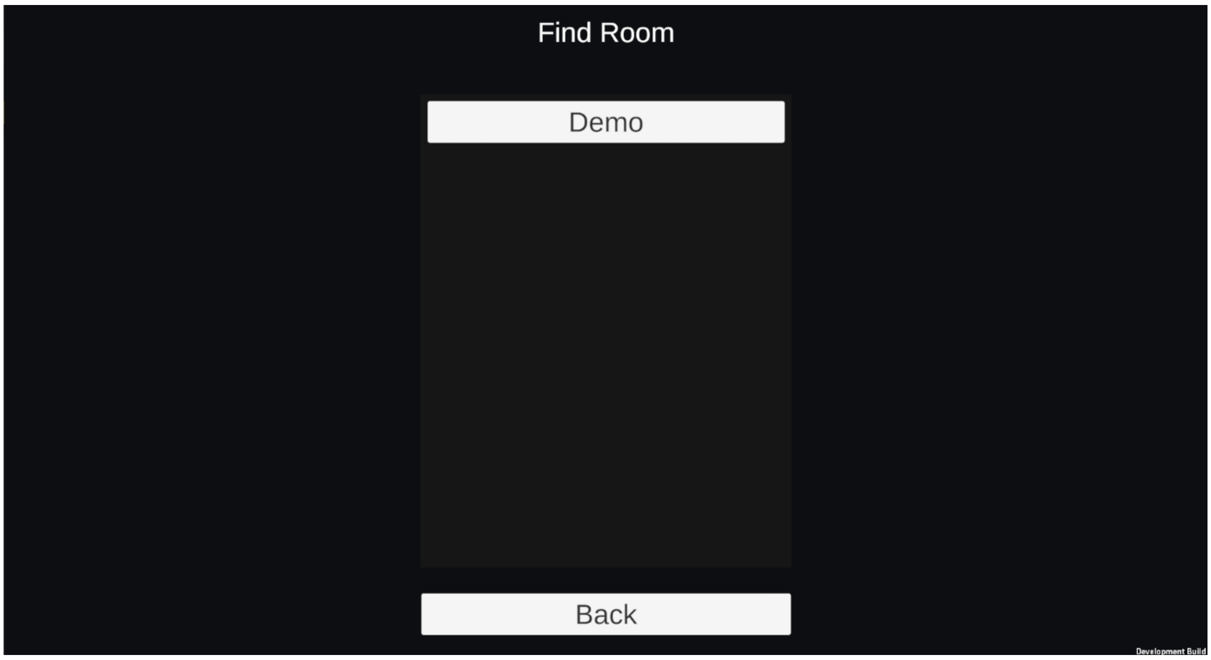


Fig. 6.1.4 Find-Room menu

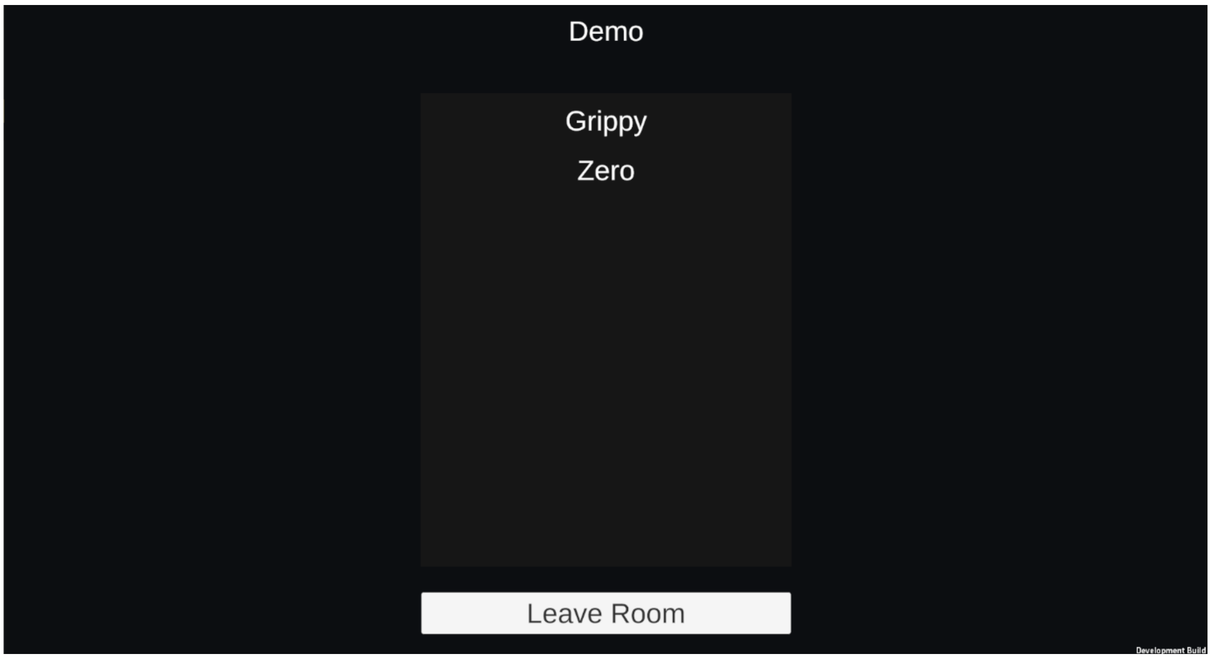


Fig. 6.1.5 Lobby

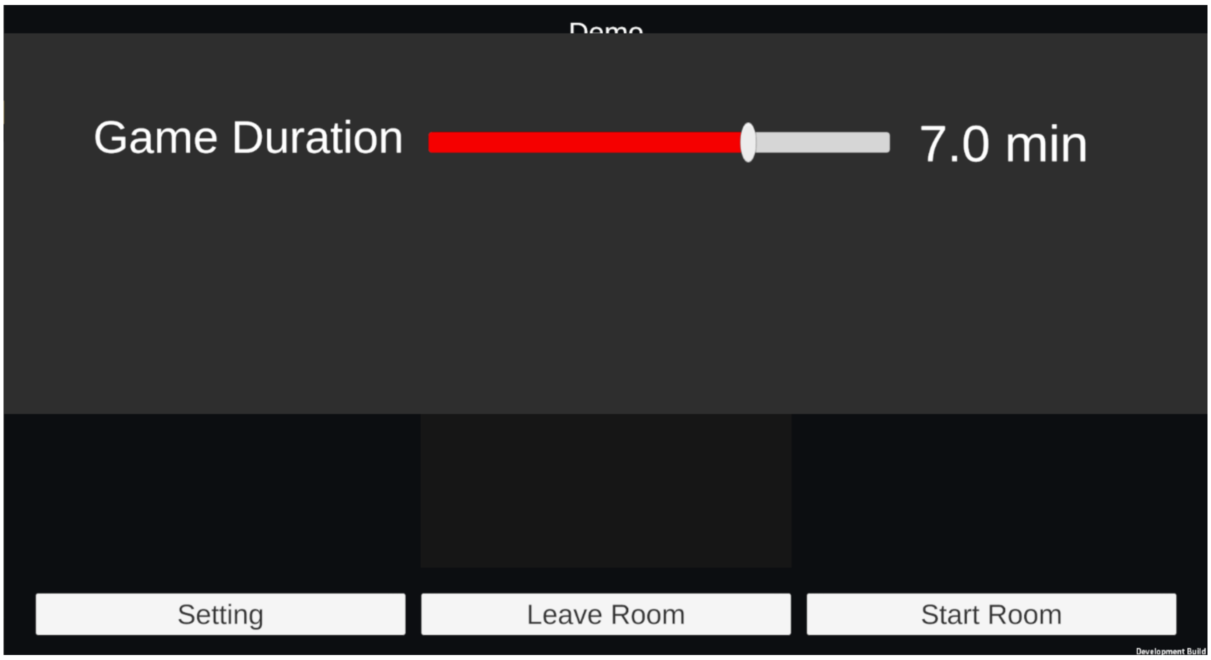


Fig. 6.1.5 Setting UI

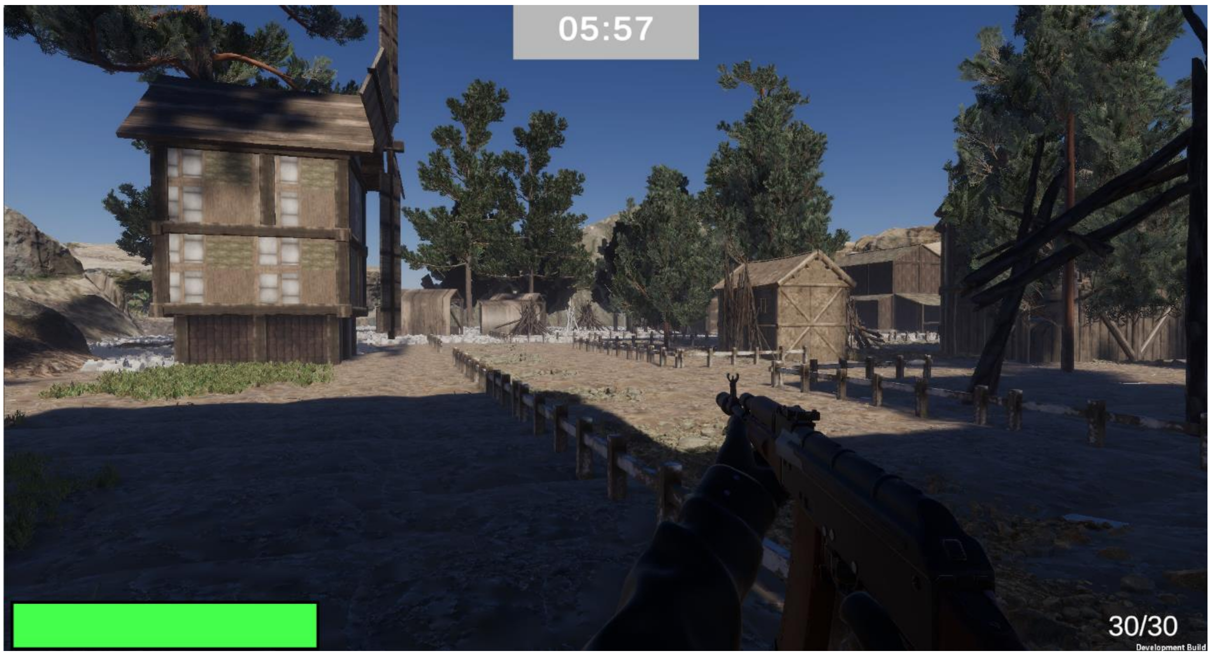


Fig. 6.1.6 Game

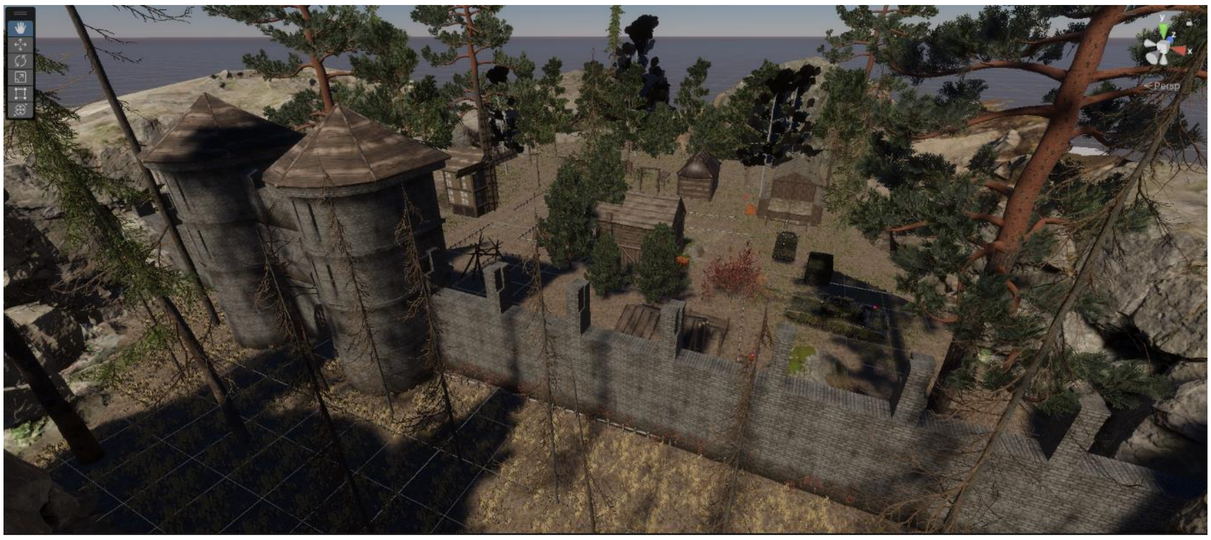


Fig. 6.1.7 Environment Scene

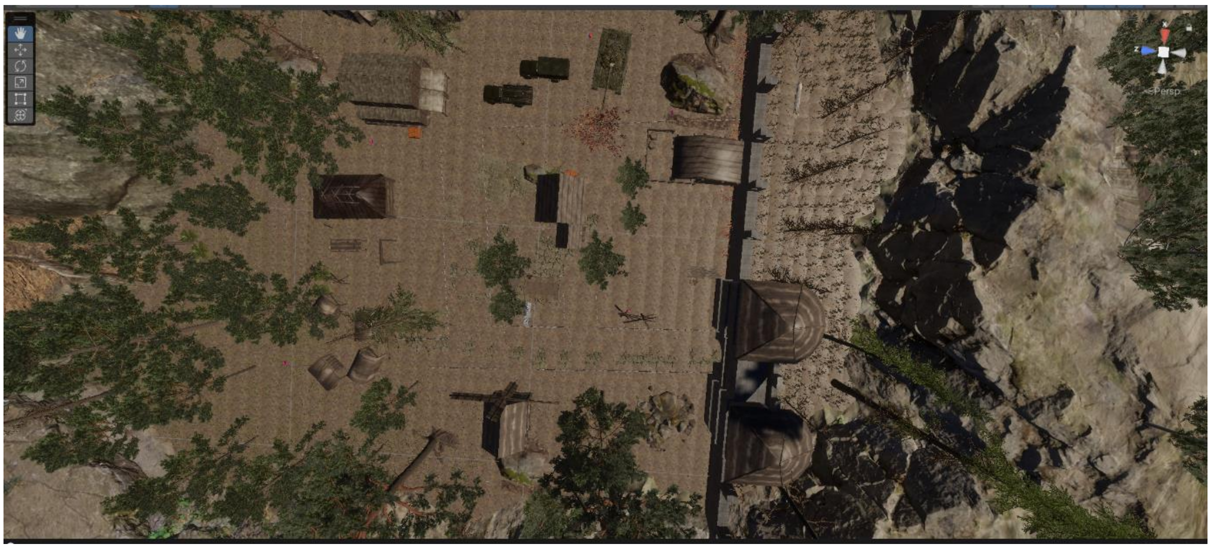


Fig. 6.1.8 Environment Scene from Top view

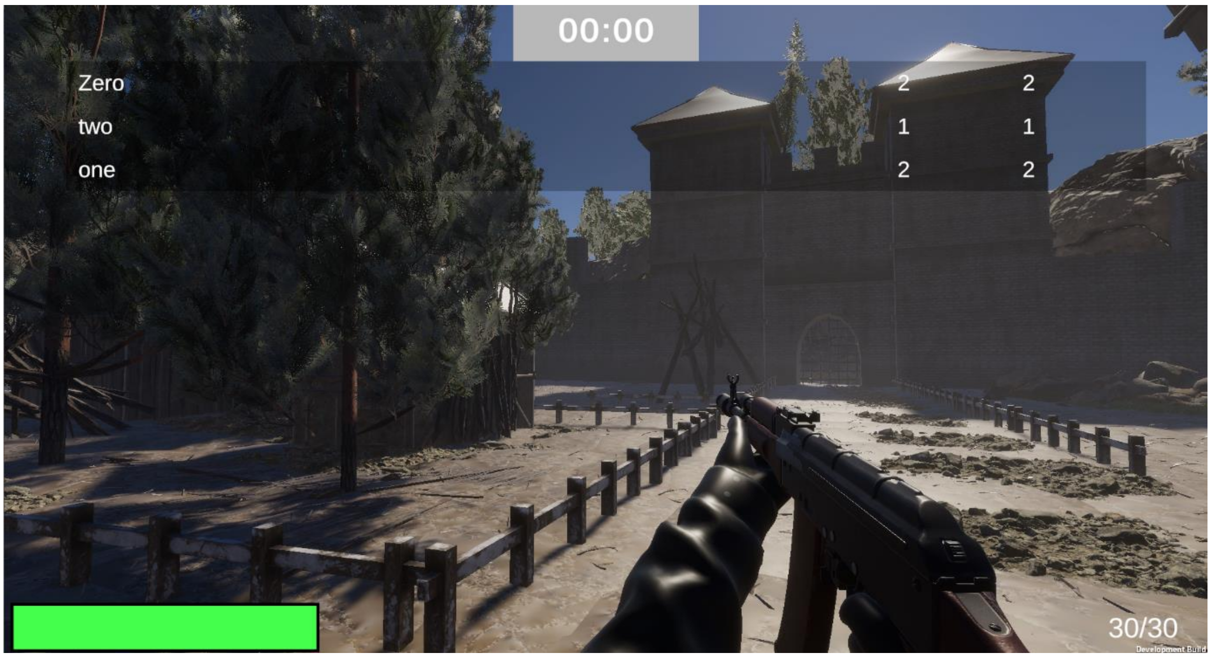


Fig. 6.1.9 Scoreboard

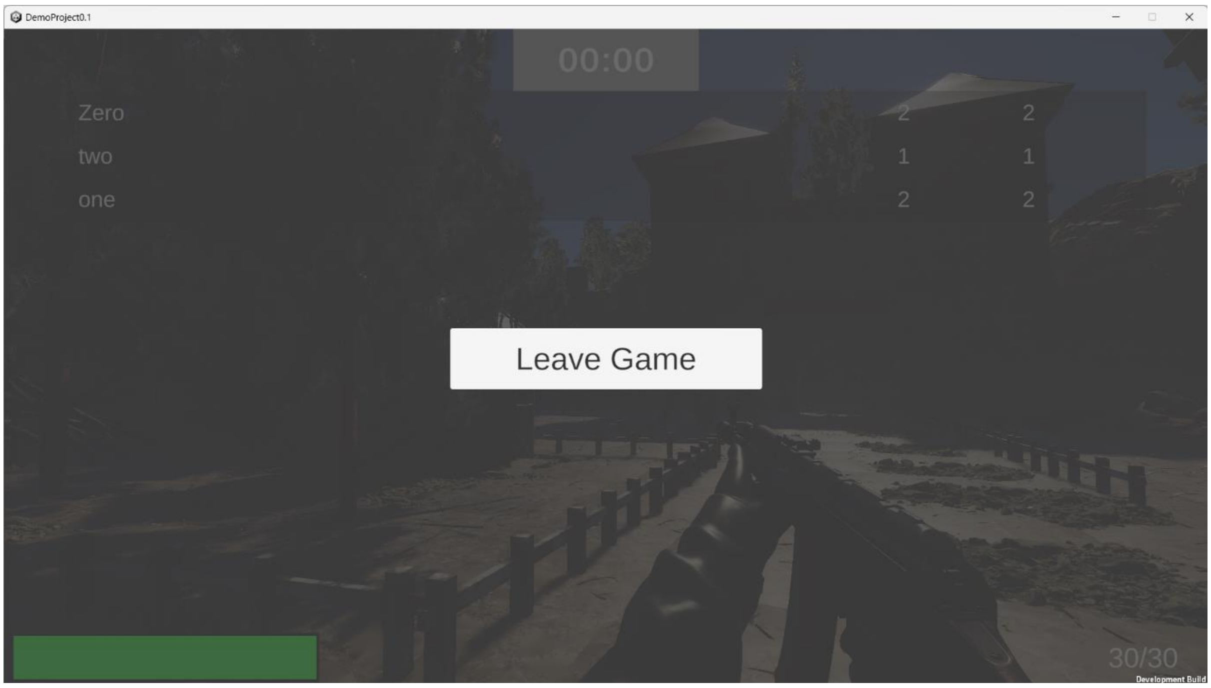


Fig. 6.1.10 Leave Game UI

CHAPTER 7
SOFTWARE TESTING

The case of an online multiplayer first-person shooter (FPS) game like ours will have elaborate and multi-dimensional testing of the software to prove that your game runs smoothly, providing an engaging user experience with no major bugs. Since it is a multiplayer FPS game, testing will have to be done at different levels including functionality, performance, security and user experience.

Functional Testing

Any game's functionality must be tested to ensure that everything works as expected, thus accrediting to the core game-play mechanics of shooting and movement not to mention collision detection and the behavior any weapon should be verified so that it works correctly and is consistent. More than that, in-game logic concerning health, damage and scoring needs checking if correct under all conditions.

Functional testing for our game also involves testing the lobby functionality where players create and join lobbies. Here we are testing many scenarios includes players can create the lobby or join the lobby that created by other player and player can leave the game after joining the lobby. The host player who created the lobby only he can start the game and he can set the game duration before the game start; other client players cannot be done these features. The game’s duration that we set before should be checked to make sure that the changes have taken effect and are saved across sessions.

Performance Testing

Performance testing is essential for all online multiplayer game, since it makes sure that the game would not reduce user experience when many players playing at once. The tests mainly focus on checking the stress on the servers, performance during peak loads and figuring out how many concurrent users the game can support before it lags or just crashes.

Latency and lag testing are also very important, as these literally affecting on the gameplay experience. Run tests through all kinds of network conditions to ensure that the game is responsive and fair for all players the player with a quality connection should not make any difference. Moreover, frame rate consistency must be monitored be sure that the game runs without errors on different hardware configurations.

Usability Testing

Usability testing is a process that typically establishes whether a game is user-friendly and fun to play. This includes examination of the UI elements, like the main menu, lobby screens, in-game HUD and leaderboards. All these UI elements require assessments in terms of use, clarity of information and responsiveness to the user's inputs.

Usability testing in both beta testing phases is invaluable with the feedback from real players. Feedback is necessary in showing how intuitive the game controls are, how easy it is to understand the game mechanics and how much fun the overall experience will be. That can then be followed up by the necessary adjustment before the final release.

Unit Testing

Unit testing is independent testing of each unit of the game. It means that every piece of code be it a specific function or method is independent of other components or modules of the game and is tested independently to check if it performs the intended functionality. These tests will find out the bugs at a very beginning stage of development, so we can make correct those bugs and as a result it would less costly before their faults spread to other areas in the game.

For example, these would include unit the logic of the players' movements, shooting mechanics, health calculations and updating the score in our game.

Integration Testing

Integration testing is about the interaction between different components or scene of the game. Individual units are unit tested after that the unit are going to be integrated and the interaction between them is tested to make sure they work together correctly. This type of testing will highlight the problems that appear when different components of the system interact with one another.

These integration tests would ensure that the lobby system interacts well with the Photon server and the game logic communication. These tests ensure the integrated system works as requested and the data flows are working well between the components.

System Testing

System testing is going to test the whole game as an integrated system. This testing is performed in order to validate whether a game confirms all specify requirements and whether it works well within the considered environment. The system testing includes all the aspects of game testing there are functionality, performance, security, usability, etc.

For instance, for system testing of your game, the entire process should be tested from start to finish it should log in, join or create a lobby, set up the game, play its match and view leaderboards. Testers should hence simulate real-world scenarios that would arise between different components and see if the game handles situations gracefully and provides experiences that are seamless for its players.

Table 7.1

Test Case ID	Test Case Description	Pre-Conditions	Test Steps	Expected Result	Status
TC001	Verify player can create a new lobby	Player is logged in	1. Navigate to main menu 2. Click on 'Create Lobby' 3. Enter lobby details 4. Click 'Create'	Lobby is created and player is taken to the lobby screen	Pass
TC002	Verify player can join an existing lobby	Player is logged in and lobby exists	1. Navigate to main menu 2. Click on 'Join Lobby' 3. Select a lobby from the list 4. Click 'Join'	Player is successfully joined to the selected lobby and taken to the lobby screen	Pass
TC003	Verify game starts correctly when all players are ready	Players are in the lobby	1. Ensure all players in the lobby are ready 2. Host clicks on 'Start Game'	Game starts and players are taken to the game scene	Pass

TC004	Verify in-game HUD displays player health and score	Game is in progress	1. Start a game 2. Observe the in-game HUD	Player's health and score are correctly displayed on the HUD	Pass
TC005	Verify player can adjust game duration from settings	Player is in the main menu	1. Navigate to settings 2. Adjust game duration using the provided UI 3. Save settings	Game duration is updated and saved correctly	Pass
TC006	Verify player can leave the game and return to the main menu	Game is in progress	1. Pause the game 2. Click on 'Leave Ga me' 3. Confirm action	Player is returned to the main menu	Pass
TC007	Verify player can shoot and register hits on other players	Game is in progress	1. Start a game 2. Aim and shoot at other players 3. Observe hit registration	Shots are registered correctly, and hits are reflected in the target player's health	Pass
TC008	Verify leaderboard is displayed correctly after the game ends	Game has ended	1. Complete a game 2. Observe the leaderboard screen	Leaderboard displays correct player rankings and scores	Pass
TC009	Verify player can change character colour	Player is in the main menu	1. Navigate to character customization 2. Select a new character colour	Character colour is successfully changed and displayed in the game	Pass

			3. Save changes		
TC010	Verify player can adjust game music volume	Player is in the main menu	1. In main menu 2. Adjust music volume slider 3. Save changes	Music volume is adjusted according to the slider setting	Pass
TC011	Verify reloading animation plays correctly	Game is in progress	1. Start a game 2. Fire weapon until reloading is triggered or We can reload if the current ammo is less the max ammo	Reloading animation plays smoothly without glitches	Pass
TC012	Verify walking animation plays correctly	Game is in progress	1. Start a game 2. Move the character forward, backward, left, right	Walking animation plays smoothly without glitches	Pass
TC013	Verify idle animation plays correctly	Game is in progress	1. Start a game 2. Stop all character movement	Idle animation plays smoothly without glitches	Pass
TC014	Verify sprinting animation plays correctly	Game is in progress	1. Start a game 2. Press the sprint button	Sprinting animation plays smoothly without glitches	Pass

TC015	Verify aiming down sights (ADS) animation plays correctly	Game is in progress	1. Start a game 2. Press the aim button to ADS	ADS animation plays smoothly and transitions correctly	Pass

CHAPTER 8
CONCLUSION

Developing an online FPS game has many challenges and complications associated with it. Completing this project has covered nearly every technical and creative domain. This project starts from making user-friendly interfaces to creating powerful multiplayer functions through Photon PUN2 such that the gaming experience becomes smooth and problem-free for the users. Functional, performance, usability, unit and integration and system testing methodologies have been through carefully for testing every component in the game. The testing of the game covered all the critical core mechanics of the game with shooting, movement, scoring and playing with multiple players over many different network conditions with totally no loss in game performance.

Completion of this project will display a huge understanding of Game development principles and the combination of various technologies and tools. This architecture shows how it works from server-client interaction to real-time data synchronization with a persistent leaderboard. This is a project that connected with requirements not only in technical experience when designing a complicated multiplayer FPS game combined with the implementation of proper game development approach from design to the final testing did not make it easier. This experience gave extremely useful understanding into game development processes and put a strong base for future projects in the gaming industry.

CHAPTER 9
FUTURE ENHANCEMENT

In the development pipeline are some very interesting enhancements that shall give actionable depth and variety to the game-playing experience of an online multiplayer first-person shooter. Allocated among these would be additional game modes. One example would be a zombie survival mode in which players are placed together in a map where they must work cooperatively to repel waves of increasingly difficult zombie enemies. This cooperative gameplay will add fresh dynamics and really encourage people to team up together.

Other planned improvements include in-game mobility. This could in turn take a leaf from games like Mirror's Edge or Assassin's Creed, where players can clamber and flow around the environment with fluidity. Graphics enhancements are also on the way to make the game more appealing further. Some of the features include improvements in texture detail, unrivaled light rendering and advanced particle effects. All these steps will increase the engagement level of the game and enhance its appearance; hence it contributes to the progress in player experience. New environmental features are planned, which shall comprise urban scenarios to desolate wastelands-each different and unique, setting up Pierce into action-packed environs.

The addition of new audio effects will increase the atmosphere of the game many times. Quite realistic shots, excellent ambient sounds and dynamic music will help in making the gameplay much more engaging. Installation of advanced sound technologies, including spatial audio, would enhance gameplay even more by giving them a more real and strategic auditory environment to play in.

The last one will be in terms of network performance and stability. This is because it will enhance the smoothness and freedom from lag that a multiplayer gameplay demands. Regular updates will address emerging issues to make sure that the game remains both a reliable and enjoyable platform for players.

APPENDIX – A

BIBLIOGRAPHY

https://www.photonengine.com/en-US/PUN

https://docs.unity3d.com/Manual/index.html

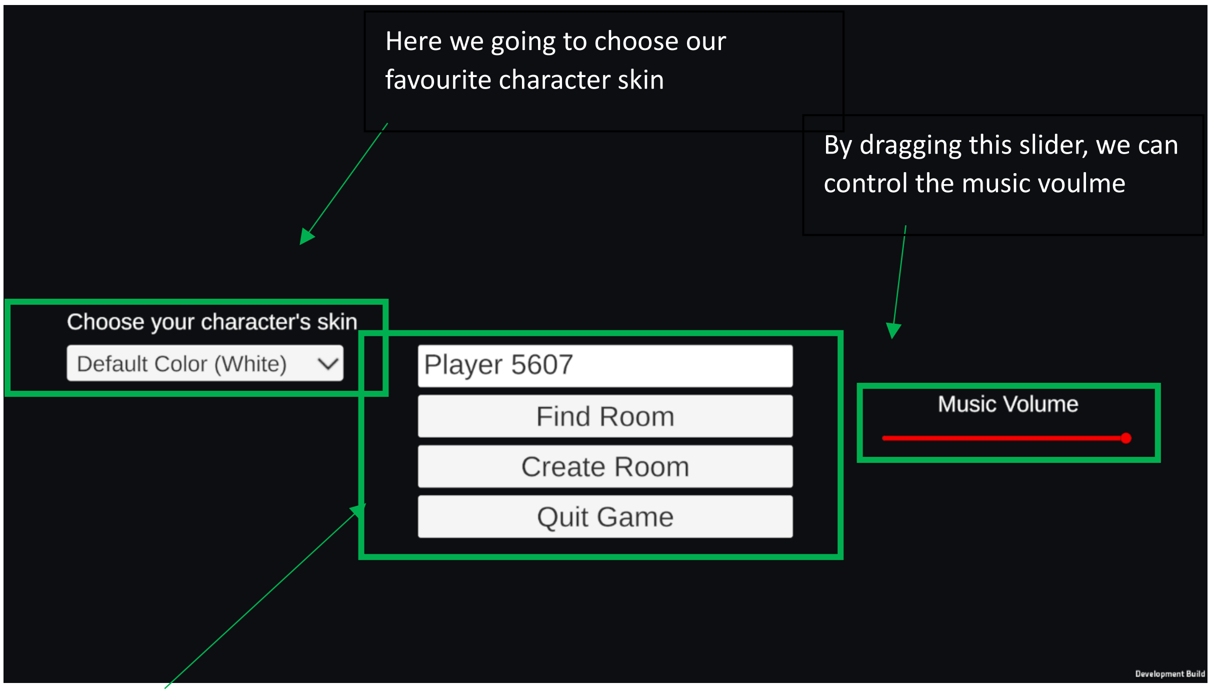
https://learn.unity.com/project/create-a-multiplayer-game

https://doc.photonengine.com/en-us/pun/v2/getting-started/pun-intro

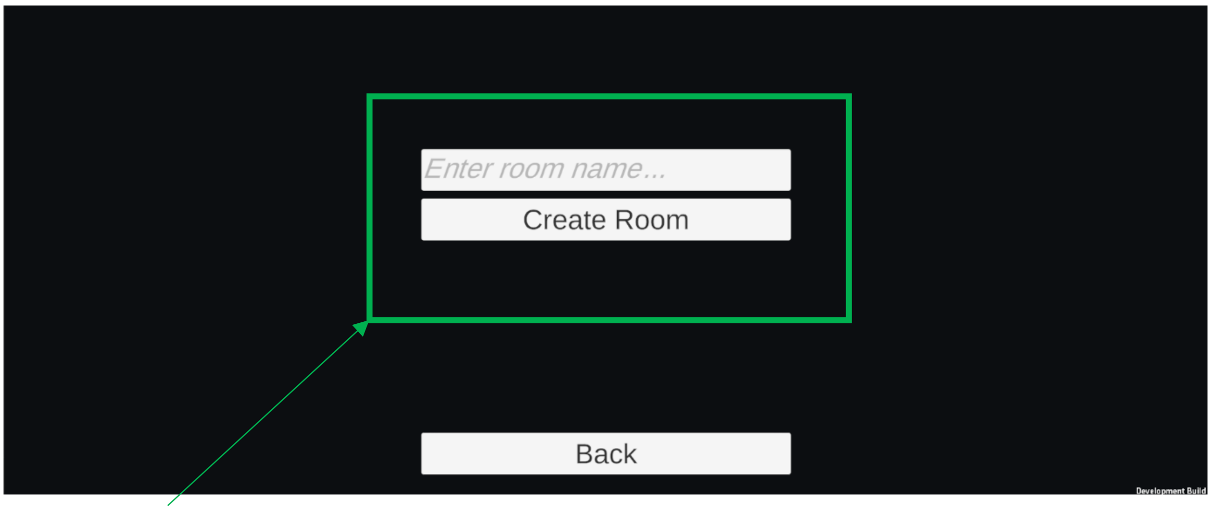
- Gregory, J. (2018). *Game Engine Architecture* (3rd ed.). A K Peters/CRC Press.
- Goldstone, W. (2015). *Unity 5.x Cookbook*. Packt Publishing.
- Smith, A. (2019). Implementing multiplayer networking in Unity using Photon PUN2. *Journal of Game Development*, 14(2), 34-50.

APPENDIX -B

USER MANUAL

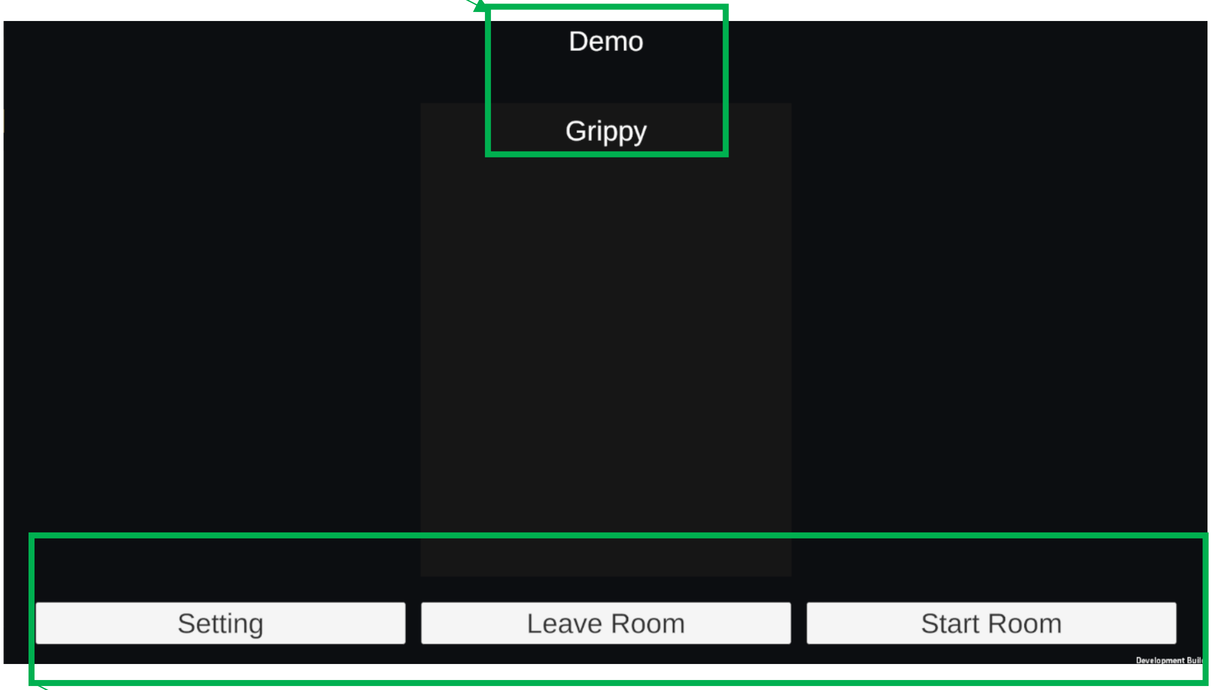

Here we going to choose our favourite character skin
By dragging this slider, we can control the music voulme
Choose your character's skin
Default Color (White)
Player 5607
Find Room
Create Room
Quit Game
Music Volume
Development Build

1. Enter your name or automatically generate
2. Click “Find Room” to search available room
3. Create a new room by Clicking “Create Room”
4. Click “Quite Game” to close the game.

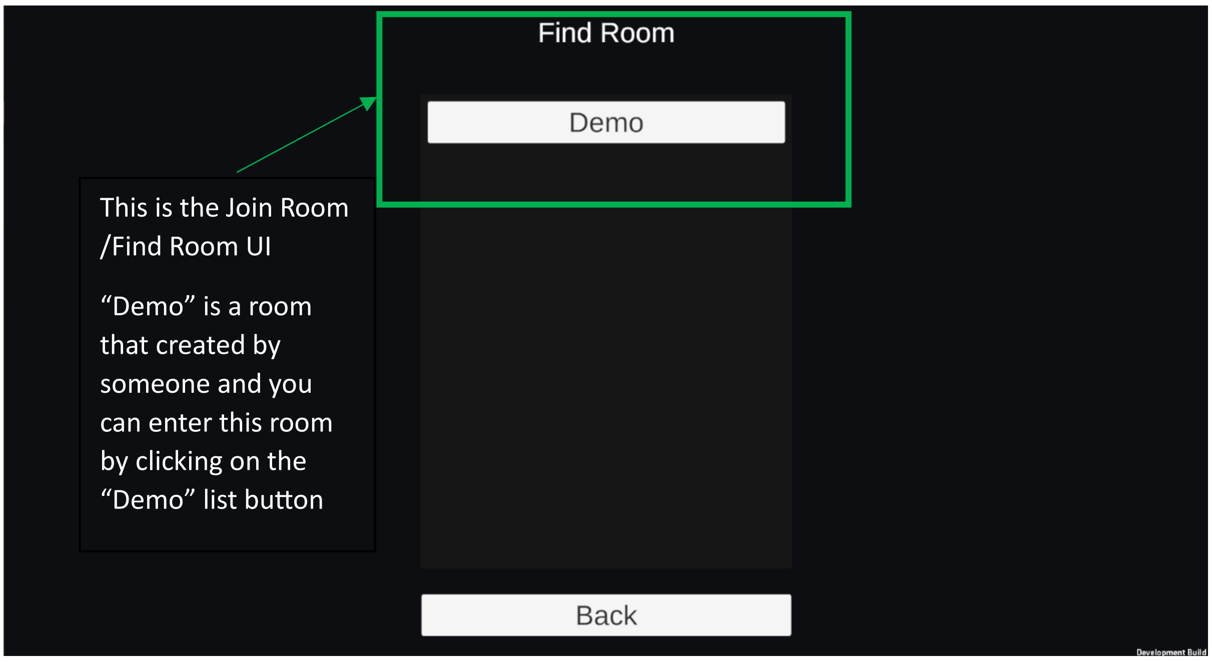


1. Write your desire room name
2. Click “Create Room” to create the room or click Back if you want to go back to main menu.

1. "Demo" is the Lobby name
2. "Grippy" is the Player name(Host) who create the room

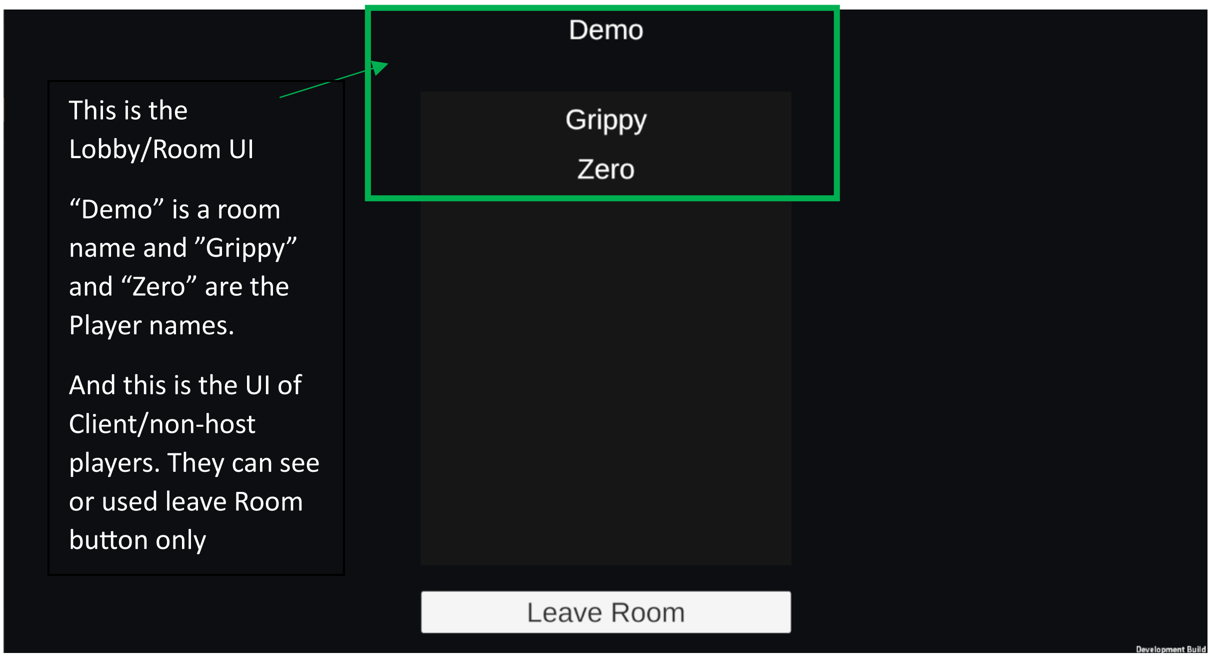


1. "Setting" button is visible and can only use by host player and it is used for setting the duration of a game.
2. Click "Leave Room" to leave the room
3. Click "Start Room" to start the game.



This is the Join Room /Find Room UI

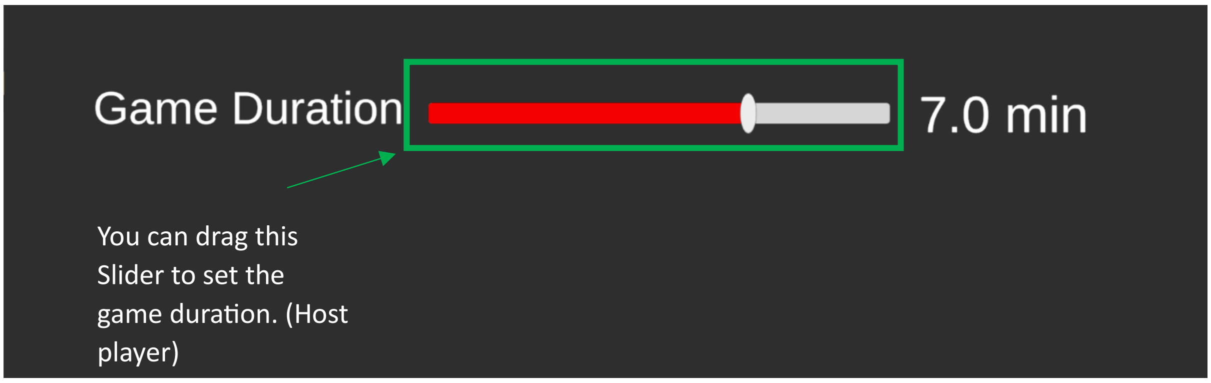
"Demo" is a room that created by someone and you can enter this room by clicking on the "Demo" list button



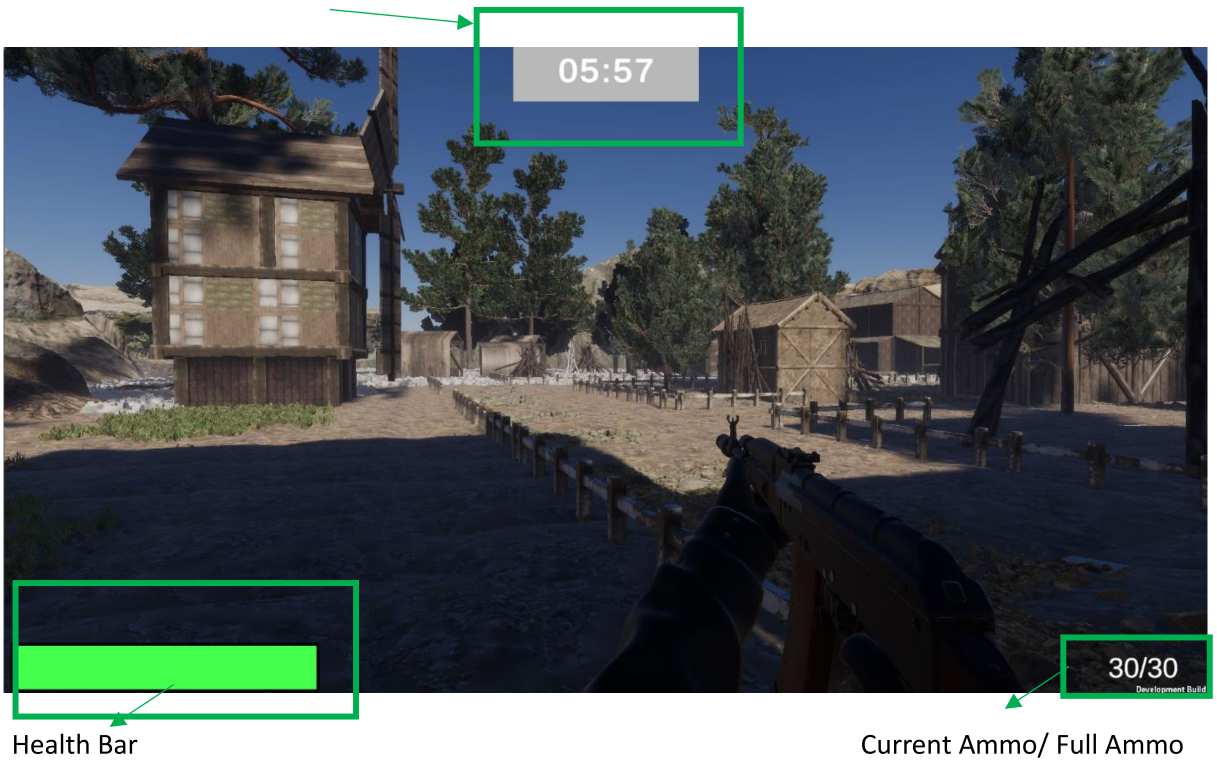
This is the Lobby/Room UI

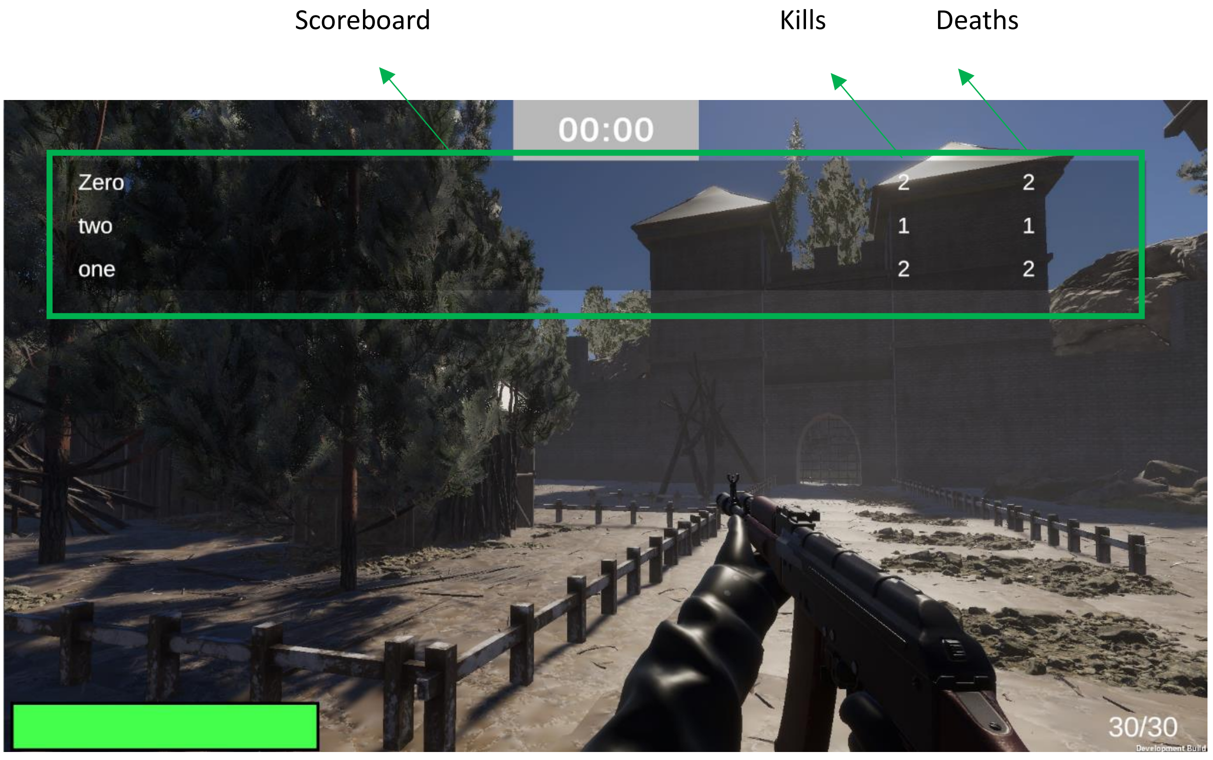
"Demo" is a room name and "Grippy" and "Zero" are the Player names.

And this is the UI of Client/non-host players. They can see or used leave Room button only



You can drag this Slider to set the game duration. (Host player)





Zero	2	2
two	1	1
one	2	2