CHAPTER 1

INTRODUCTION

1.1. Introduction

Multiplayer First-Person Shooters (FPS) have emerged as a dominant force in the gaming landscape, captivating millions of players worldwide with their adrenaline-pumping action and intense competition. These games transport players into immersive virtual worlds where they assume the role of a skilled combatant, engaging in fast-paced firefights against both human and AI opponents. With a focus on precision aiming, tactical maneuvers, and quick reflexes, FPS games demand sharp skills and strategic thinking.

The allure of multiplayer FPS lies in its ability to foster intense competition and camaraderie among players. Whether it's cooperating with teammates to achieve a common objective or engaging in fierce one-on-one duels, these games offer a unique social experience. From classic modes like Deathmatch and Capture the Flag to more complex objectives, multiplayer FPS cater to a wide range of player preferences. The constant evolution of the genre, with advancements in technology and game design, ensures that there's always something new and exciting to discover [1].

Multiplayer FPS have become synonymous with esports, attracting massive audiences and prize pools. Professional players showcase their exceptional abilities on a global stage, inspiring aspiring gamers to hone their skills. The competitive nature of these games, coupled with the thrill of victory and the sting of defeat, creates an addictive experience that keeps players coming back for more.

1.2. Problem Statement

Multiplayer First-Person Shooters (FPS) are often resource-intensive games, requiring powerful hardware to deliver smooth gameplay and high-quality visuals. This can pose challenges for players with older or less capable systems. Additionally, the competitive nature of these games can not work on different kinds of Operating System

1.3. Objective

The objectives of the Multiplayer First-Person Shooters game are:

- To run game in cross platform devices.
- To run game using low utilization of resources.

1.4. Scope and Limitation

Every Video Games have its own unique features and its limitations. This videogame offers following scope and limitation:

1.4.1 Scope

- Multiplayer Support: It supports Multiplayer gameplay experience.
- Lobby System: It supports Lobby System to join muliplayer game.
- Low end Device: It supports on Low End Devices.

1.4.2 Limitation

- User Experience: This videogame lacks user experience.

1.5. Report Organization

Report organizing for each chapter that has been documentation is refers to specific format and it is easy to understand by the readers for the whole of the report.

Chapter 1: Introduction

It provides an overview of the project's significance, problem statement, objectives, scope, limitations, and development methodology.

Chapter 2: Background Study and Literature Review

It covers foundational knowledge and a review of related projects and research. It includes the study of the current trends, preferences of people, the existing systems.

Chapter 3: System Analysis and Design

It discussed about the System analysis, Requirement analysis (Functional and Non-Functional), Feasibility analysis (Economical, Technical, Operational, Schedule) and Data Modeling , process Modeling, System design(Architectural Design, Database schema Design, Physical DFD) that has been used during the development of the system and system design.

Chapter 4: Implementation and Testing

It details the tools used, module implementations, and testing procedures.

Chapter 5: Conclusion and Future Recommendations

It consists of conclusion on the project development and future Recommendation.

CHAPTER 2

BACKGROUND STUDY AND LITERATURE REVIEW

2.1 Background Study

The roots of the First-Person Shooter (FPS) genre can be traced back to the early 1970s.

However, it wasn't until the mid-1990s that games like Wolfenstein 3D and Doom truly defined the genre with their first-person perspective, weapon-based combat, and level exploration. The natural progression was to introduce multiplayer elements, initially through split-screen and LAN play. The internet's rise transformed multiplayer FPS into a dominant force, with games like Quake and Unreal Tournament focusing primarily on online competition [2].

Multiplayer FPS games typically revolve around skill-based combat, emphasizing movement, aiming, weapon handling, and map knowledge. Teamwork is often crucial for success in objective-based modes. These games have significantly influenced the gaming industry, driving technological advancements and contributing to the growth of esports. They've also impacted other media, with FPS-inspired action becoming commonplace in movies and television.

Multiplayer FPS games have faced challenges such as maintaining game balance, combating cheating, and ensuring fair competition. The industry continues to evolve, with developers exploring new mechanics, virtual reality, and augmented reality to keep players engaged.

2.2 Literature Review

Counter-Strike: Global Offensive (CS:GO) is a multiplayer tactical first-person shooter that has solidified its position as a cornerstone of the esports industry. Building upon the foundation laid by its predecessors, CS:GO offers intense, competitive gameplay centered around two teams, Counter-Terrorists and Terrorists, engaging in a battle of wits and reflexes. With its emphasis on tactical maneuvers, economic management, and precise gunplay, CS:GO has captivated millions of players worldwide, fostering a thriving professional scene and a dedicated community of enthusiasts [3].

Call of Duty: Modern Warfare marked a significant reboot of the iconic franchise, delivering a gritty and realistic portrayal of modern warfare. The game introduced a new visual engine, enhancing immersion and player engagement. With a focus on intense multiplayer action, a compelling campaign, and a cooperative Special Ops mode, Modern Warfare redefined expectations for the series. Its emphasis on tactical gameplay, combined with a strong emphasis on storytelling, contributed to its critical and commercial success, solidifying its place as a major contender in the FPS genre [4].

CHAPTER 3

SYSTEM ANALYSIS AND DESIGN

3.1 System Analysis

System analysis of Multiplayer First-Person Shooters game is a process of studying the software application to understand its components, interactions, and how they work together to achieve the application's goals. This analysis can be used to improve the application's performance, reliability, and security.

The system analysis of Multiplayer First-Person Shooters game can be divided into two main phases: functional analysis and structural analysis. Functional analysis focuses on understanding the application's features and how they are used by users. Structural analysis focuses on understanding the application's components and how they interact with each other.

The results of the system analysis can be used to improve the application in a number of ways, such as improving performance, reliability, and security. The system analysis of Multiplayer First-Person Shooters game is a valuable tool for improving the application's quality and performance.

3.1.1 Requirement Analysis

Requirement analysis for a multiplayer FPS game involves identifying the network-related functionalities and performance expectations necessary for a seamless multiplayer experience.

i. Functional Requirements

- Multiplayer Game: Users can play Multiplayer game with the help of network.
- Lobby: Users can create and join lobby in Multiplayer First-Person Shooters game.

Use Case Diagram

On Multiplayer First-Person Shooters Videogame Users can Login. Users can view Dashboard and Select Map for gameplay. Users can Create or Join Lobby with other

users. User can kill enemy.

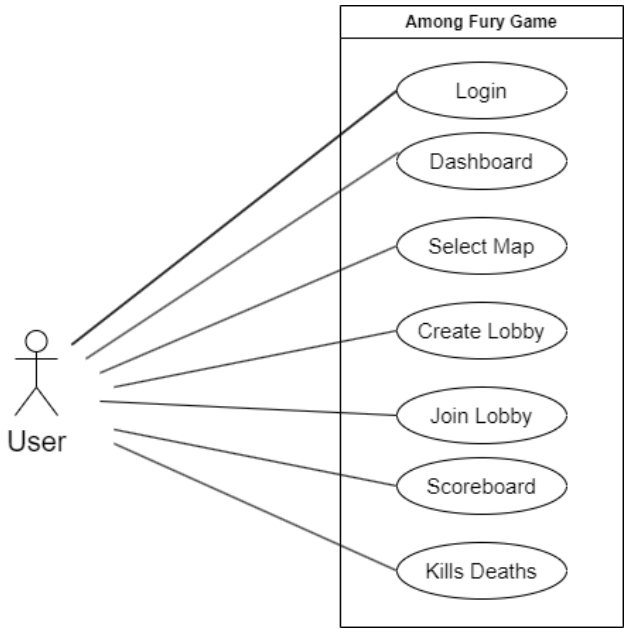

Figure 3.1: Use case Diagram of Videogame

On Web Users can Register and Login. Users can Add Confirm and Delete Request from other users. Users can send and receive Message from other users.

Admin can view Dashboard.

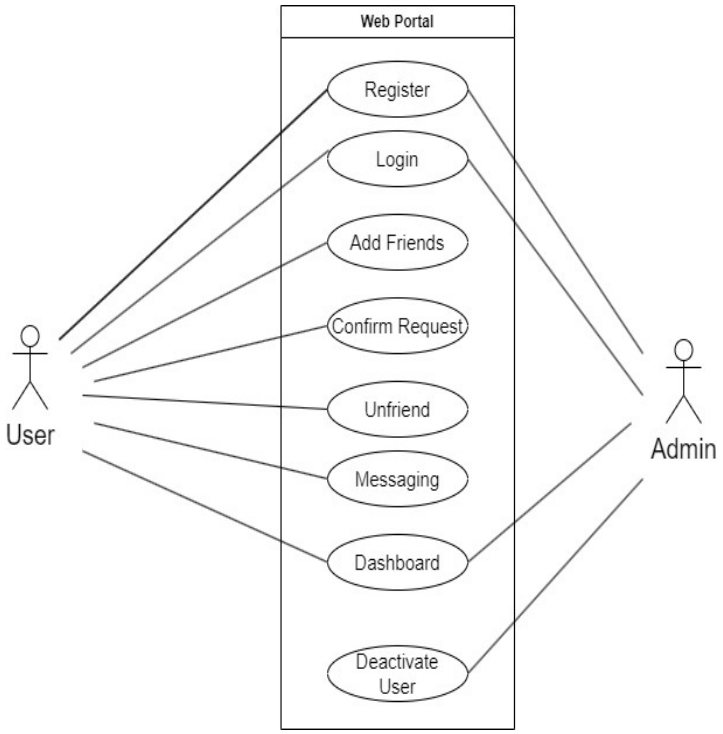

Figure 3.2: Use case Diagram of Web page

ii. Non-functional Requirements

- Scalability: This Multiplayer First-Person Shooters game can handle multiple users simutaneously.
- Network Synchronization: This Multiplayer First-Person Shooters game can play from multiple devices.

3.1.2 Feasibility Analysis

i. Technical Feasibility

This project uses existing web technologies and tools so there is no difficulty in developing this project.

ii. Operational Feasibility

This project is simple and intuitive, using this application doesn't require extra knowledge. It is very user-friendly and can be easily adapted by any new user that will use this game. As it uses well-known network-related technologies, it is easy to maintain.

iii. Economic Feasibility

This project is very cost-friendly as it uses existing free technologies like Unity, Blender, etc. There is no cost for the tools that are being used for its development. The only thing is to be done is making an environment for the development with an effective supervision.

iv. Schedule

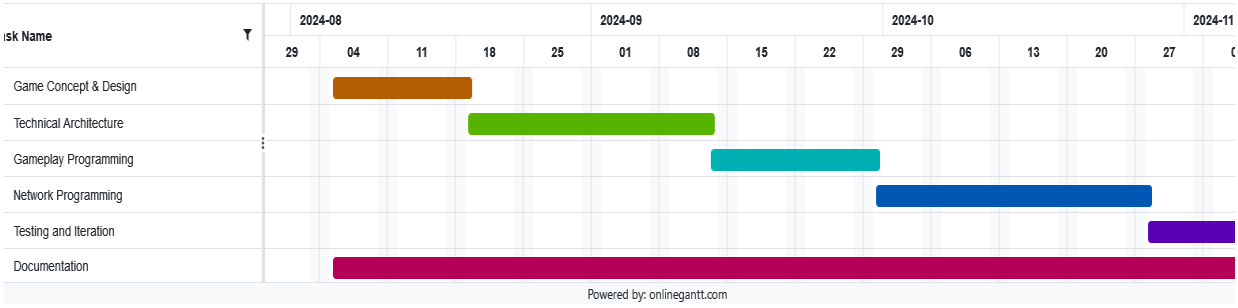

Figure 3.3: Gantt Chart for Multiplayer First-Person Shooters

The working schedule of the proposed system will be including the first 2 weeks of Game Concept and Designing and another next 3 weeks of Technical Architecture of importing modules. The Gameplay Programming will expect to be completed in another 2 weeks and 3 weeks of Network Programming and 2 weeks of testing as well. All system design, development and testing will be done interchangeably. Documentation of the project will be done starting from system design until the testing ends.

3.1.3 Data Modelling (ER-Diagram)

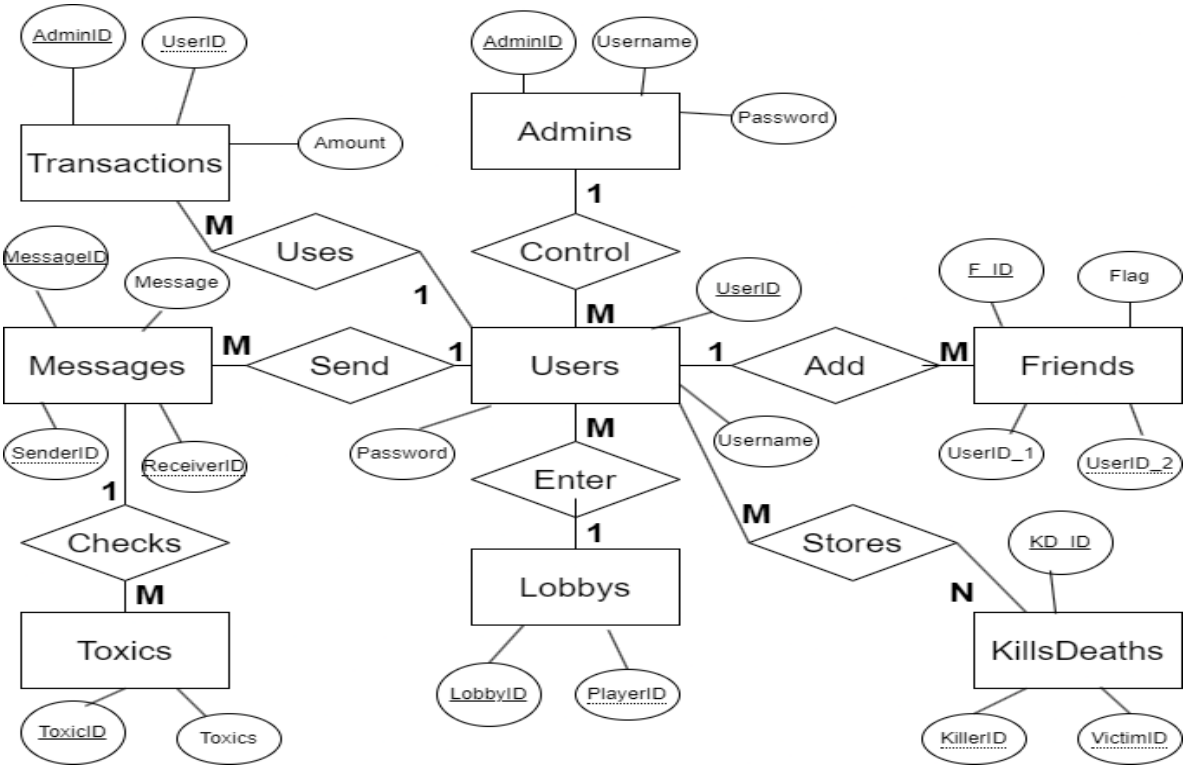

Figure 3.4: Data Modelling (ER-Diagram) of Multiplayer First-Person Shooters

This is figure of ER diagram in which show the relation between entities. A user is identified by user id, Name(username), Password, etc. The user can sent multiple message which is identified by message ID, message, sender and receiver. A user can send friend request to friend which can be accepted. The User can enter Lobby and join it. The User is controlled by Admin.

3.1.4 Process Modelling (DFD)

Users can play Multiplayer FPS game on their computer. Multiplayer FPS game interacts with Web for backend connectivity. Admin also can interact with Web for user management.

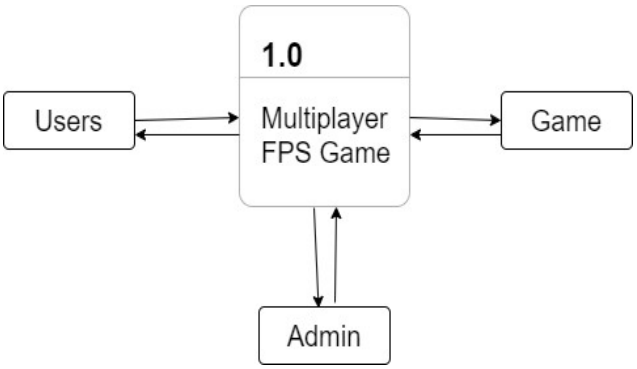

Figure 3.5: Process Modelling (Level 0 DFD) of Multiplayer FPS game

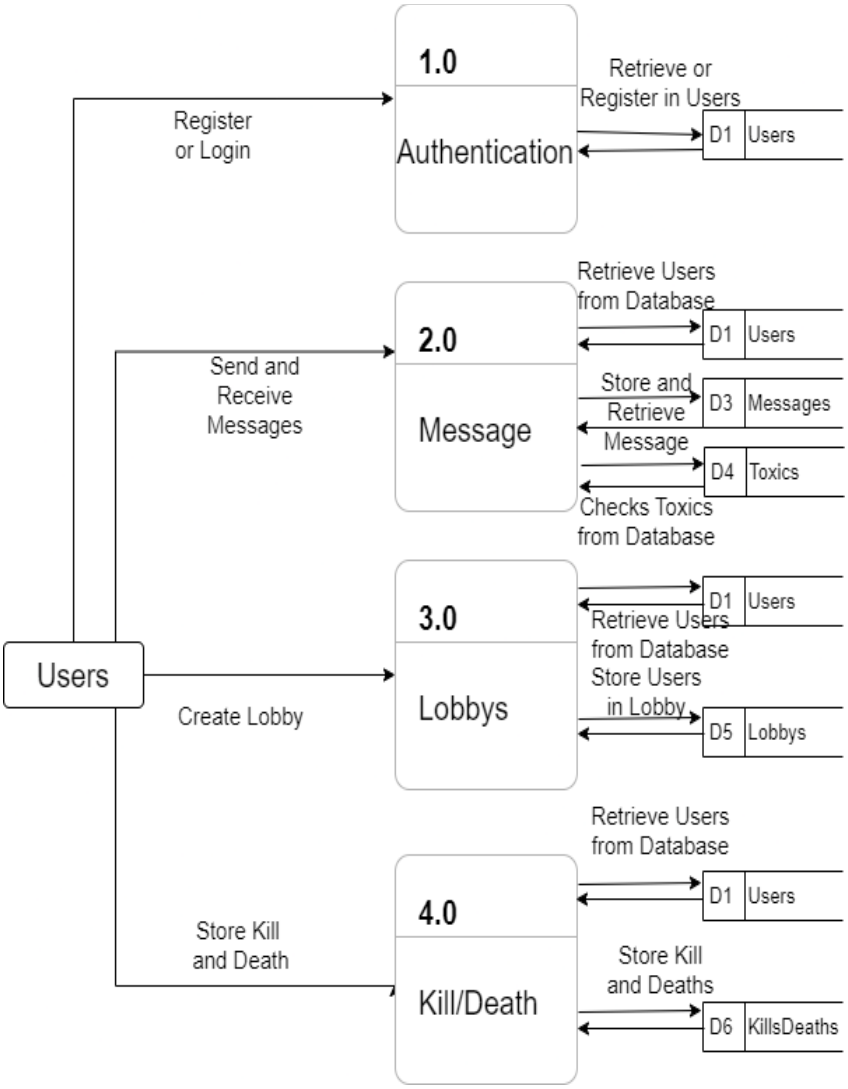

Figure 3.6: Process Modelling (Level 1 DFD) of Videogame

Users can Authenticate with Database on Video game. Users can send and receive Messages. Users can Create and Join Lobby on Video game. User can kill enemy. User can interact in Microtransactions.

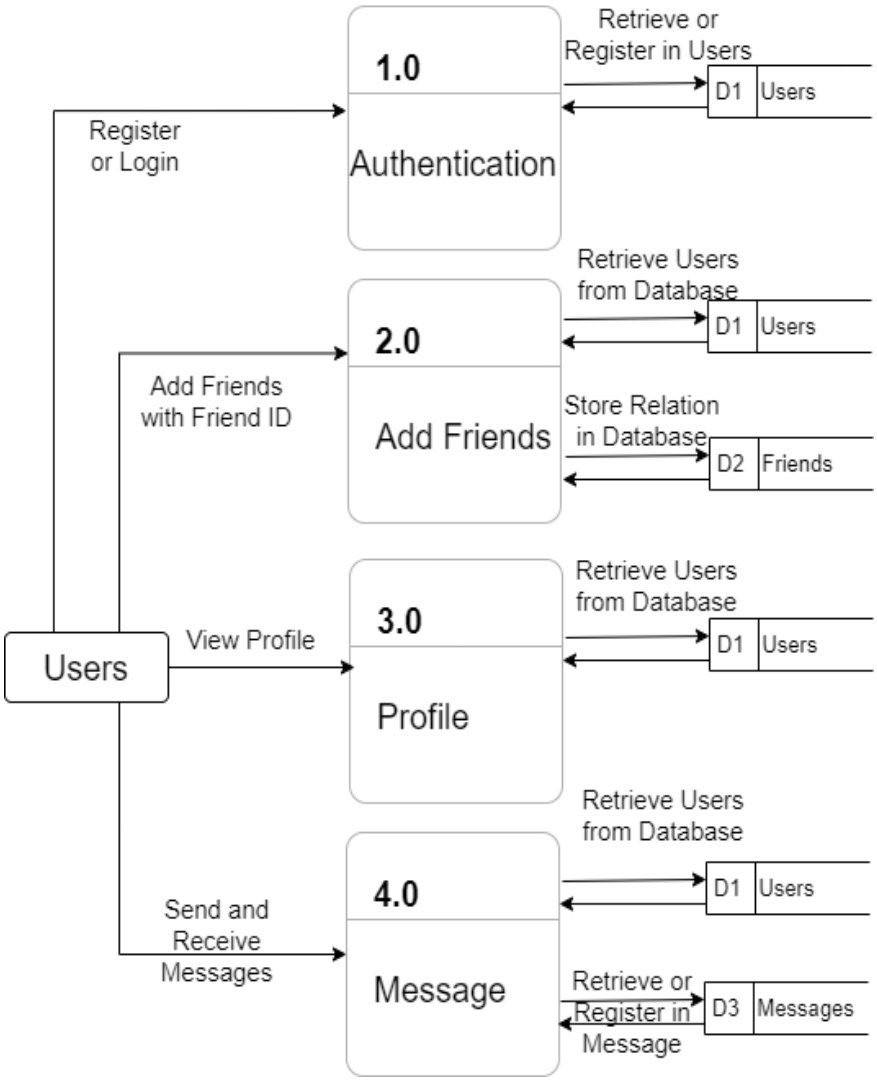

Figure 3.7: Process Modelling (Level 1 DFD) of Web page of User

On Web Users can Register and Login through the form. Users can Add, Confirm and Delete Friends. User can View own or users Profile. Users can send and receive messages with Friends.

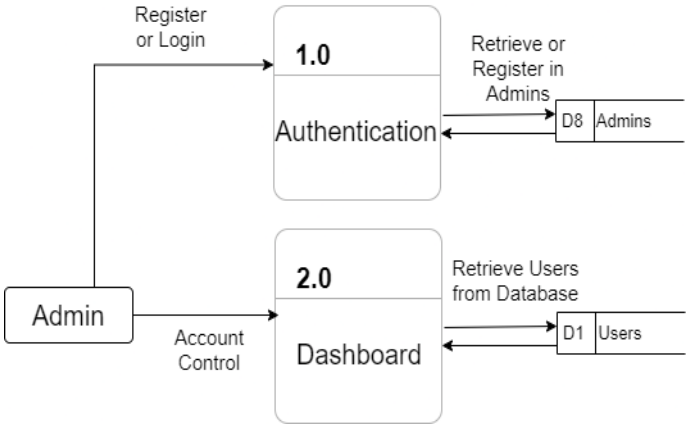

Figure 3.8: Process Modelling (Level 1 DFD) of Web page of Admin

Admin can Login through the Web. Admin can control users through the Dashboard. Admin can Confirm the Micro-Transactions used by user.

3.2 System Design

3.2.1 Architecture Design

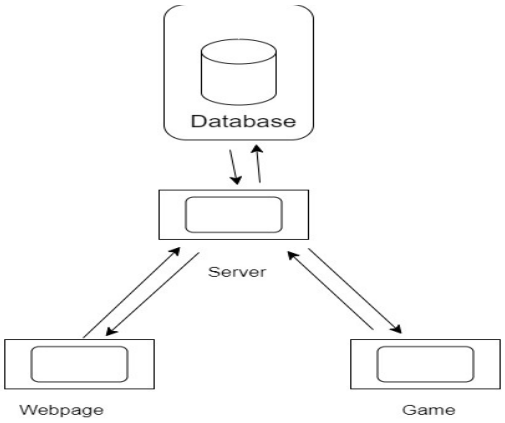

Figure 3.9: Architecture Design of Multiplayer First-Person Shooters

In Multiplayer First-Person Shooters videogame, Users interact with Videogame that interact with backend with the help of API. Similarly the Webpage for videogame also interacts with same backend. The backend process the data and stored in database or respond data from database to webpage or videogame.

3.2.2 Database Schema Design

This is the Database Schema Design of Multiplayer First-Person Shooters in which user table contains id, username, password, etc. Messages contains sender and receiver username. Lobby creates the lobby data. Friends stores the friend lists. KillDeaths store the kill and deaths of the player. Transaction stores the micro transactions which is controlled by admin. Toxic stores all toxic word.

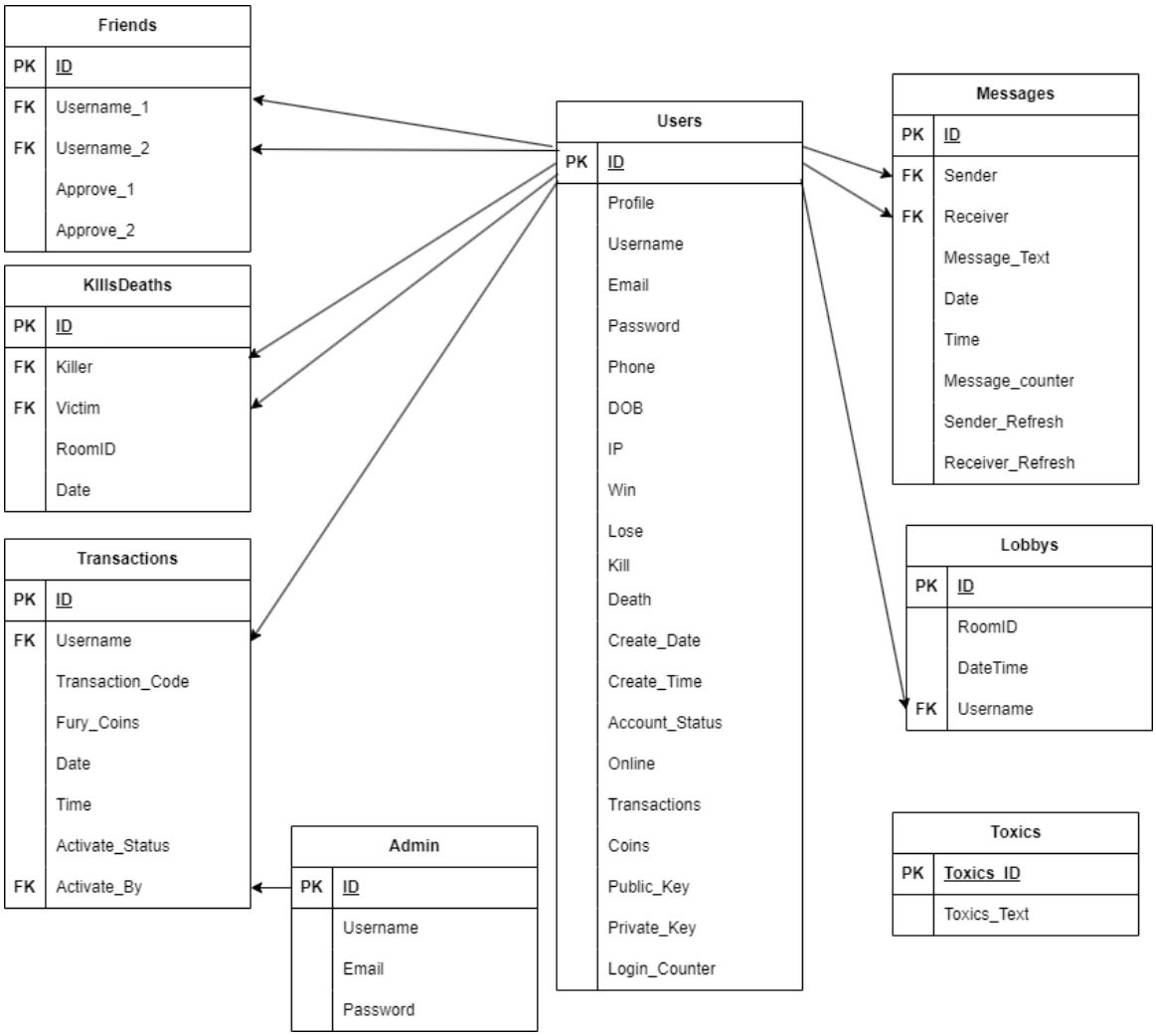

Figure 3.10: Database Schema Design of Multiplayer First-Person Shooters

3.2.3 Interface Design (UI Interface / Interface Structure Diagram)

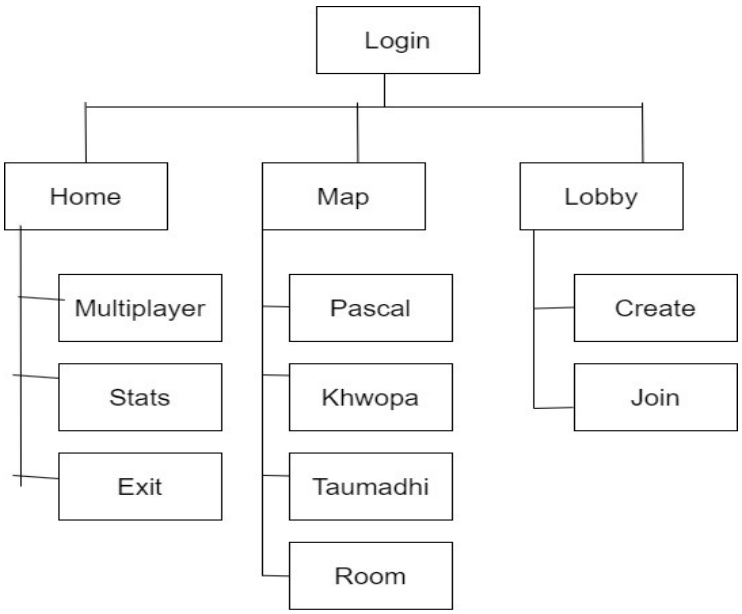

Figure 3.11: Interface Structure Design of Multiplayer First-Person Shooters

Before implementing actual design, an Interface Design are constructed to visualize User interaction with system as they browse Login and redirect to Home. The interface design will closely follow our Functional Decomposition Diagram shown above.

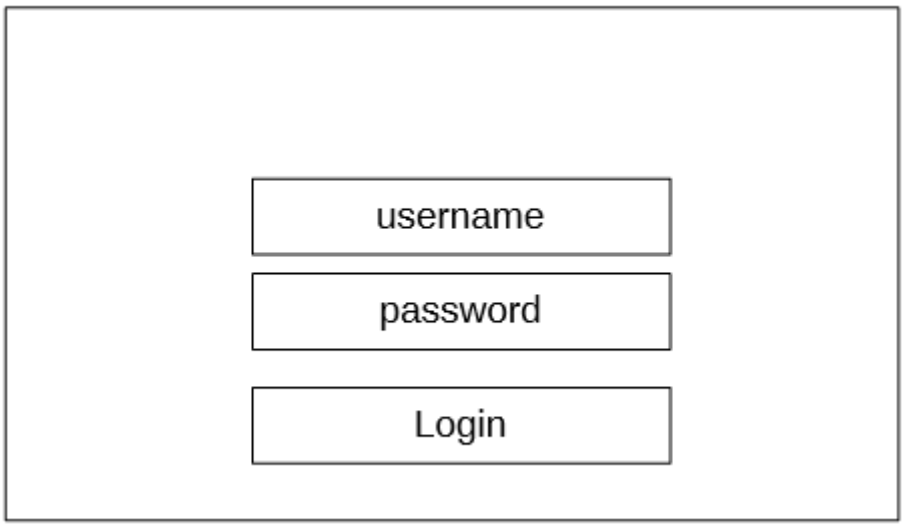

Figure 3.12: Wireframe Design for Login of Multiplayer First-Person Shooters

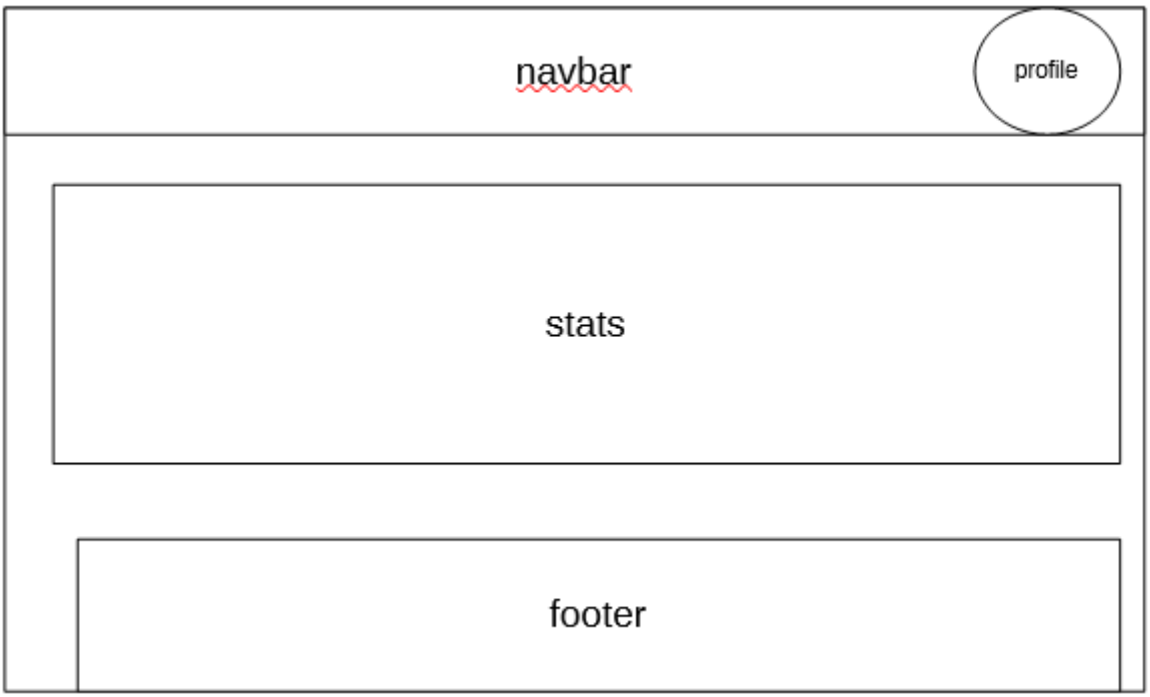

Figure 3.13: Wireframe Design for Home of Multiplayer First-Person Shooters

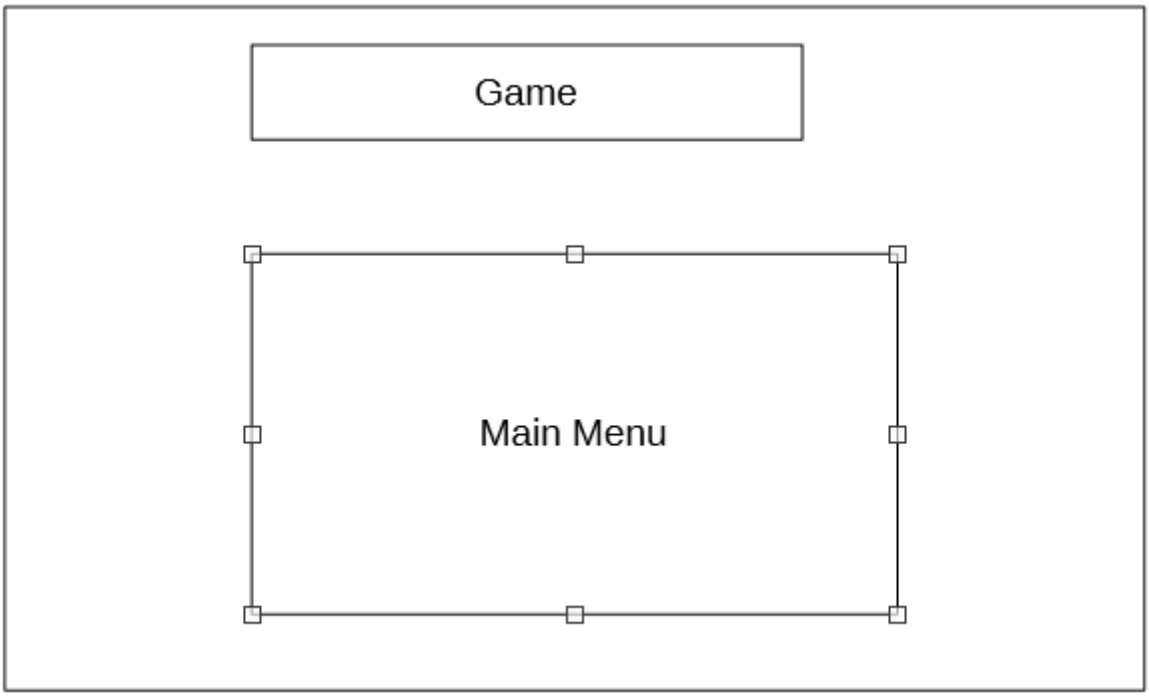

Figure 3.14: Wireframe Design for Menu of Multiplayer First-Person Shooters

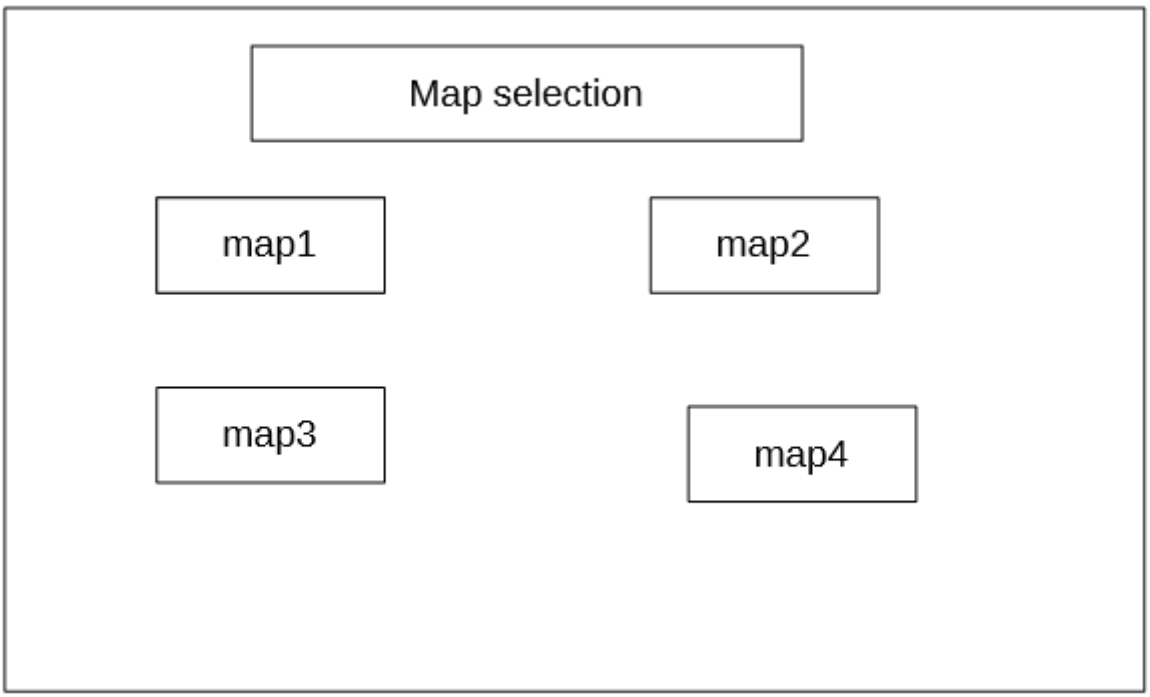

Figure 3.15: Wireframe Design for Map of Multiplayer First-Person Shooters

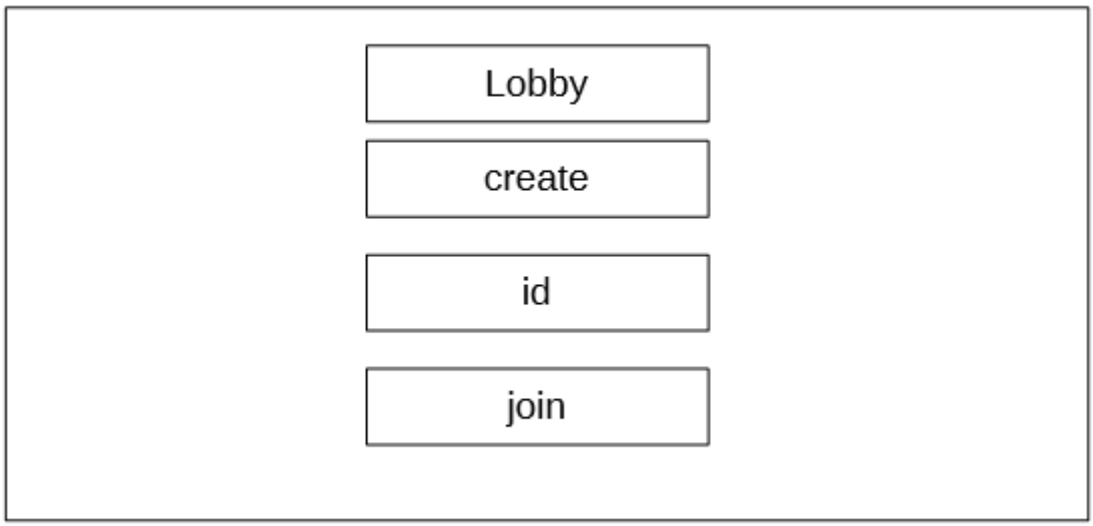

Figure 3.16: Wireframe Design for Lobby of Multiplayer First-Person Shooters

3.3 Algorithm details

Network Synchronization

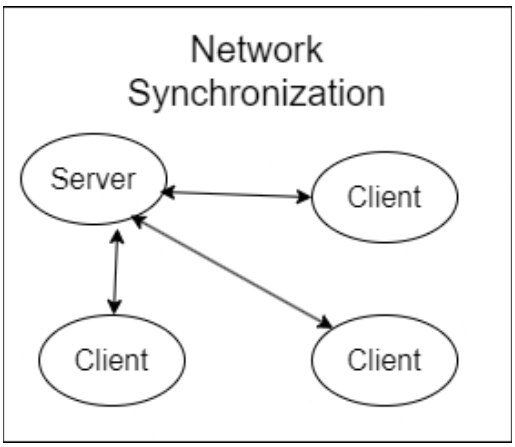

Figure 3.17: Algorithm of Network Synchronization

Step 1: Initialization

- Each client starts with a local game state (e.g., positions of objects, player stats).
- The server initializes the authoritative game state and tracks the state for all clients.

Step 2: Client State Updates

Each client performs the following steps:

- Input Handling: The client gathers user input (e.g., movement, action) and sends it to the server in the form of a "command" or "action" packet.
- Local Simulation: While waiting for the server response, the client simulates the result of the input locally (client-side prediction) to provide immediate feedback to the player.
- Send to Server: The input or state change is sent to the server along with a timestamp or sequence number.

Step 3: Server Processing

The server performs the following steps:

- Receive Updates: The server receives updates (inputs or state changes) from clients.

- Apply Updates: The server applies the inputs to the authoritative game state.
- State Validation: The server checks whether the received inputs are valid and applies them to the game world.
- Send Updates: The server periodically sends the latest authoritative state to all clients.

Lobby

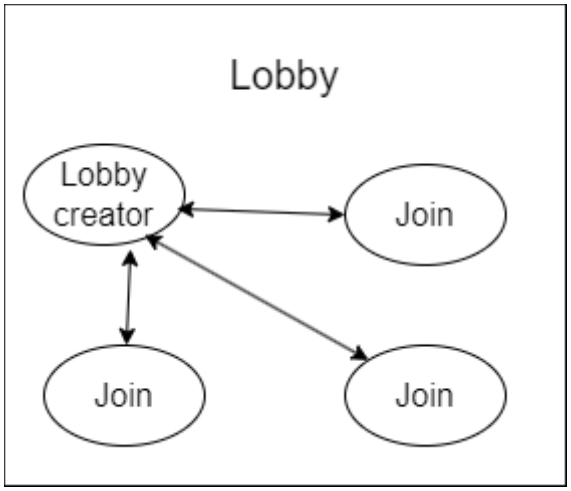

Figure 3.18: Algorithm of Lobby System

Step 1: Host Creates a Lobby

- Input: Player A (host) selects to create a lobby.
- Actions:
    1. Host sends a "Create Lobby" request to the server.
    2. Server creates a unique lobby ID and initializes the lobby settings (e.g., max players, game mode).
    3. Server assigns the host as the lobby leader.
    4. Server responds to host with lobby details (ID, current state, settings).
    5. The lobby is now available for other players to join.
- Output: Lobby created and available.

Step 2: Players Join Lobby

- Input: Players (e.g., Player B, Player C) request to join the lobby.

- Actions:
    1. Player sends a "Join Lobby" request with the lobby ID.
    2. Server checks if the lobby is still open and if the player is allowed to join (e.g., max players reached).
    3. If successful, the server adds the player to the lobby.
    4. Server broadcasts the new player’s presence to the existing players in the lobby.
- Output: Players successfully join the lobby and see the list of players in the lobby.

CHAPTER 4

IMPLEMENTATION AND TESTING

4.1 Implementation

Implementation basically means the phase where the system is actually being built. First, all the information that gathered is studied and analyzed and implemented a system in operation for users. Implementation usually consists of coding, testing, installation, documentation, training and support.

4.1.1 Tools Used

Tools

- **VS code**

  Vs code is used for writing code for the project. It is used for the coding HTML, CSS, Javascript and C#.

- **Unity Engine**

  Unity Engine is used to build Multiplayer First-Person Shooters.

- **Browser**

  Browser like firefox is used to run the web page.

- **Blender**

  Blender is used to model the character and maps.

Programming Language

- **React**

  React is used to create frontend for the design of the website

- **CSS**

  CSS is used to design the Multiplayer First-Person Shooters. It helps to design the format, color etc.

- **Node Js**

  Node JS is used to run the backend of the program. It is used to optimise the code, connect database with code etc.

Database platform

- **MongoDb**

  MongoDb database is used to store data, retrive and display data request by user.

4.1.2 Implementation Details of Modules

The proposed system is composed of different modules such as Network Synchronization, Lobby creation, Map selection, Kill Death, Scoreboard, weapon change. etc. Implementation applicatin of this scale requires lot of resource and explaining the whole implantation process will not be clarified. So some of the modules are listed below.

- **Network Synchronization**

  The user interacts in the game can be interacts in other user game using real time network synchronization.

```
public class NetworkSync : NetworkBehaviour
{
   // Start is called before the first frame update
   void Start()
   {
      PlayerController playerController = new PlayerController();
      OnServerRPCNetworkSync(playerController);
   }
   // Update is called once per frame
   void Update()
   {

   }
   // [ServerRPC]
   public void OnServerRPCNetworkSync(PlayerController playerController)
   {
      OnObserverRPCNetworkSync(playerController);
   }
   // [ObserverRPC]
   public void OnObserverRPCNetworkSync(PlayerController playerController)
   {
      playerController.enabled = true;
   }
}
```

Listing 4.1: Code Snippet for Network Sync of Multiplayer First-Person Shooters

- **Lobby Creation**

  Lobby is created when host create a room. Other user joins the lobby using lobby id.

```
    public void CreateLobby()
    {
        lobbySuccess.text = "Lobby is Created";
        string generate_lobby_id = UnityEngine.Random.Range(10000000,
99999999).ToString();
        lobbyInput.text = generate_lobby_id;

        String host_username = username;
        string game_mode = "Full DeathMatch";
        String game_map = map;
        StartCoroutine(CreateLobbyCoroutine(generate_lobby_id, game_mode, game_map,
host_username));
    }

    // Button Join
    public void JoinLobby()
    {
        string lobby_id = lobbyInput.text;
        string player_user_id = "123";
        string player_username = username;
        StartCoroutine(JoinLobbyCoroutine(lobby_id, player_user_id, player_username));
    }
```

Listing 4.2: Code Snippet for Lobby of Multiplayer First-Person Shooters

- **Map selection**

  Map selection is done when a user create a map. Only selected map is rendered in game.

```
if (mapTaumadhi.activeSelf)
{
    map = "Taumadhi";
}
if (mapPascal.activeSelf)
{
    map = "Pascal";
}
if (mapKhwopa.activeSelf)
{
    map = "Khwopa";
}
if (mapRoom.activeSelf)
{
    map = "Room";
}
```

Listing 4.3: Code Snippet for Map of Multiplayer First-Person Shooters

- **Kill Death**

  When a user points crosshair to other player, it calls remote procedure call that reduces the other player health. Hence player is dead and respawn in other location.

```
// Kills
public static void SetKills(int clientID, int kills)
{
   instance.SetKillsServer(clientID, kills);
}

[Server]
public void SetKillsServer(int clientID, int kills)
{
   SetKillsObserver(clientID, kills);
}

[ObserversRpc]
private void SetKillsObserver(int clientID, int kills)
{
   instance._playerCards[clientID].SetKills(kills);
}


// Deaths
public static void SetDeaths(int clientID, int deaths)
{
   instance.SetDeathsServer(clientID, deaths);
}

[Server]
public void SetDeathsServer(int clientID, int deaths)
{
   SetDeathsObserver(clientID, deaths);
}

[ObserversRpc]
private void SetDeathsObserver(int clientID, int deaths)
{
   instance._playerCards[clientID].SetDeaths(deaths);
}
```

Listing 4.4: Code Snippet for KillDeath of Multiplayer First-Person Shooters

- **Scoreboard**

  When scoreboard is displayed in canvas, it record all the kill and death of the player.

```
if (Input.GetKeyDown(KeyCode.Tab))
{
   scoreboard.SetActive(true);
}
if (Input.GetKeyUp(KeyCode.Tab))
{
   scoreboard.SetActive(false);
}
```

Listing 4.5: Code Snippet for Scoreboard of Multiplayer First-Person Shooters

- **Weapon change**

  When a user changes the weapon, the only selected weapon is rendered in player. Other weapon is disabled in scripts.

```
// AWM
if (Input.GetKey(KeyCode.Alpha1))
{
   Debug.Log(isAndroid);
   InitializeWeapon(0);
}
// Pistol
if (Input.GetKey(KeyCode.Alpha2))
{
   InitializeWeapon(3);
}
// AR15
if (Input.GetKey(KeyCode.Alpha3))
{
   InitializeWeapon(1);
}
// Vector
if (Input.GetKey(KeyCode.Alpha4))
{
   InitializeWeapon(2);
}
```

Listing 4.6: Code Snippet for Weapon of Multiplayer First-Person Shooters

4.2 Testing

Testing is done as unit testing in the project where data is manually tested.

4.2.1. Unit testing

Unit testing is done by manually testing by entering value given by user. The some module of tested in Multiplayer First-Person Shooters are Login module, Register module, Add friend module and Message module.

a. Registration Testing

ID	Test Case	Test Data	Expected Outcome
U_REG_1	User Entering Data	Username: sushan Email: sushan@gmail.com Password: sushan Mobile: 98765432120 DOB: 2001/06/07	Registration Success, Redirect to Login Form.

Table 4.1: Registration Testing of Multiplayer First-Person Shooters

b. Login Testing

ID	Test Case	Test Data	Expected Outcome
U_LOG_1	User Entering Wrong Data	Username: sushan Password: password	Prompt Invalid Username or password
U_LOG_2	User Entering Correct Data	Username: sushan Password: sushan	Redirect to Home

Table 4.2: Login Testing of Multiplayer First-Person Shooters

c. Friend Page

ID	Test Case	Test Data	Expected Outcome
U_FRN_1	User sent Friend Request	User click Add Friend	Request is sent
U_FRN_2	User accept Friend Request	User click Accept	Friend is shown in list
U_FRN_3	User unfrined Friend Request	User click unfriend	Friend is deleted in list

Table 4.3: Friendlist of Multiplayer First-Person Shooters

c. Message Page

ID	Test Case	Test Data	Expected Outcome
U_MSG_1	User send message	Message: Hi	Message is sent
U_MSG_2	User send message with toxic word	Message: hey mula	Message is sent with message text “hey m***”

Table 4.4: Message Testing of Multiplayer First-Person Shooters

b. Login Testing of Admin

ID	Test Case	Test Data	Expected Outcome
A_ADM_1	User Entering Wrong Data	Username: admin Password: admin	Prompt Invalid Username or password
A_ADM_2	User Entering Correct Data	Username: admin Password: password	Redirect to Dashboard

Table 4.5: Admin Login Testing of Multiplayer First-Person Shooters

4.2.1. System testing

System testing is done by evaluating functionality, performance and reliability of Multiplayer First-Person Shooters. The some functionality of tested in Multiplayer First-Person Shooters are Network Synchronization, Lobby creation, Map selection, Kill Death, Score, Weapon change.

a. System Testing of Multiplayer First-Person Shooters

ID	Test Case	Expected Outcome
U_SYS_1	User create Lobby	Lobby is created.
U_SYS_2	User joins Lobby	User joins the game
U_SYS_2	User selects map	Map is displayed in game
U_SYS_2	User interacts with environment	Other player can view using Network Synchronization
U_SYS_2	User kill player	User respawn in other location
U_SYS_2	User view score board	Scoreboard is displayed in canvas
U_SYS_2	User change weapon	Weapon is changed in player

Table 4.6: System testing in Multiplayer First-Person Shooters

CHAPTER 5

CONCLUSION AND FUTURE RECOMMENDATION

5.1 Lesson Learnt /Outcome

The primary goal of this project was to create the Multiplayer First-Person Shooters where users can play video game with each their friends. The important lesson learnt was management of time according to complexity of the system component i.e. know which components to prioritize.

- Learn how to do system analysis and feasibility study.
- Learnt to integrate the API with server.
- Learnt how to perform Network Synchronization.
- Learnt how to create Lobby.
- Learn how to model the Character design and Maps.

5.2 Conclusion

Overall Among Fury is a Multiplayer First-Person Shooters that allows user to play video game with other users in real time. It can work on any low end device if compiled. Among Fury also creates Lobby to play with friends in a network.

5.3 Future Recommendation

Here are some of the features that can be added to increase its usability, user experience and portablility of this platform.

- Friendly user experience.
- Transaction.
- Multiple room creation.

REFERENCES

[1] Gameopedia, "The Evolution of First Person Shooter (FPS) Games," 30 April 2023. [Online]. Available: https://www.gameopedia.com/evolution-of-first-person-shooter-fps-games. [Accessed 05 August 2024].

[2] Broadband Search, "Online Gaming Statistics," 18 April 2024. [Online]. Available: https://www.broadbandsearch.net/blog/online-gaming-statistics. [Accessed 06 August 2024].

[3] RedBull, "From esports fever to cheating… Counter-Strike: Global Offensive had it all," 27 March 2024. [Online]. Available: https://www.redbull.com/int-en/history-of-counterstrike. [Accessed 06 August 2024].

[4] Call of Duty, "Call of Duty: Modern Warfare," 25 December 2023. [Online]. Available: https://www.callofduty.com/blog. [Accessed 07 August 2024].